CS 1675: Intro to Machine Learning

Neural Networks

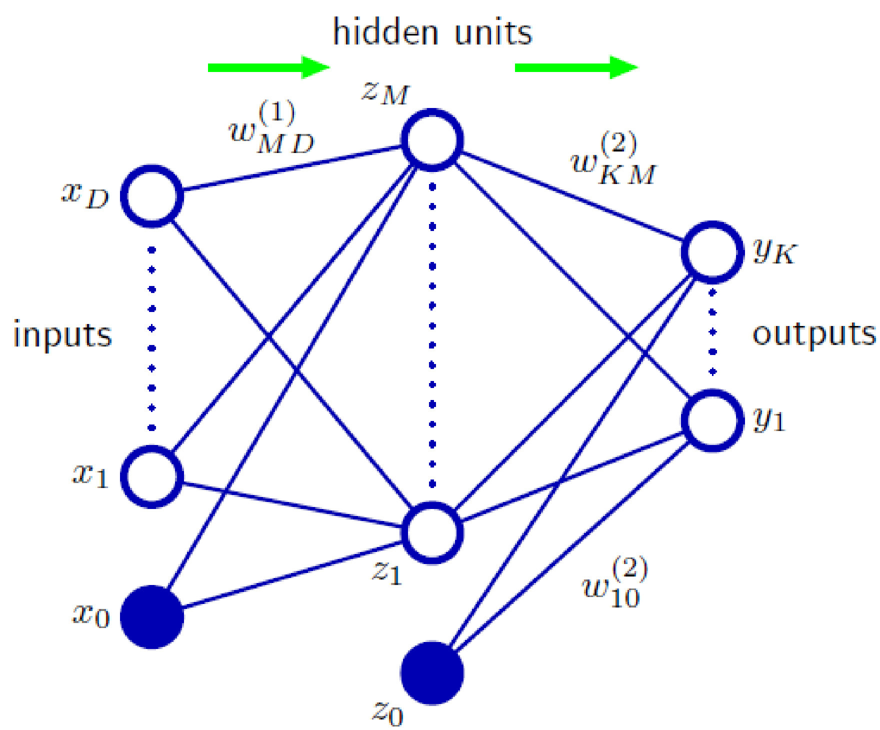
Prof. Adriana Kovashka
University of Pittsburgh
November 1, 2018

Plan for this lecture

- Neural network basics
  - Definition and architecture
  - Biological inspiration
- Training
  - Loss functions
  - Backpropagation
  - Dealing with sparse data and overfitting
- Specialized variants (briefly)
  - Convolutional networks (CNNs) – e.g. for images
  - Recurrent networks (RNNs) – for sequences, language

Neural network definition

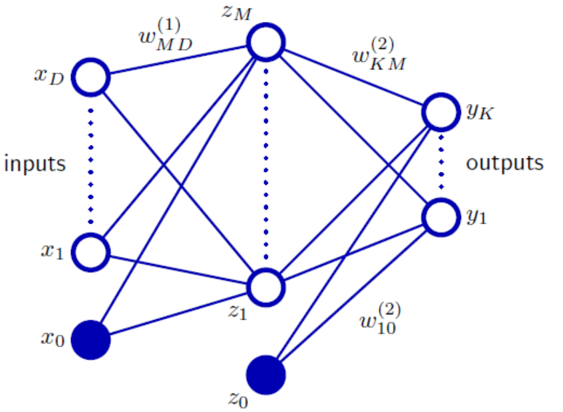
Figure 5.1 Network diagram for the two-layer neural network corresponding to (5.7). The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links coming from additional input and hidden variables x_0 and z_0. Arrows denote the direction of information flow through the network during forward propagation.



- Activations: $$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$
- Nonlinear activation function *h* (e.g. sigmoid, tanh, RELU): $z_j = h(a_j)$ $$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Figure from Christopher Bishop

Neural network definition



- Layer 2

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

- Layer 3 (final)

$$a_k =$$

- Outputs

(binary) $$y_k = \sigma(a_k) = \frac{1}{1 + \exp(-a_k)}$$

(multiclass) $$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

- Finally:

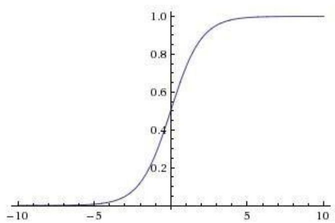
(binary)

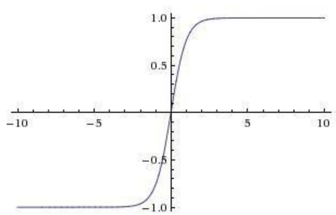
$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

Activation functions

Sigmoid

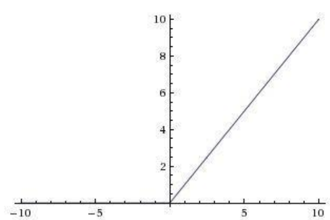
$$\sigma(x) = 1/(1+e^{-x})$$





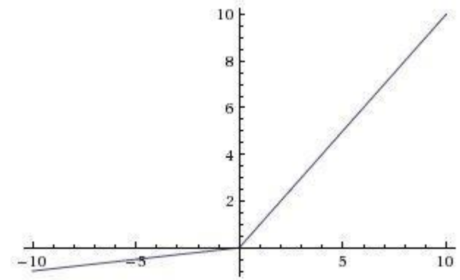
tanh tanh(x)

ReLU max(0,x)



Leaky ReLU

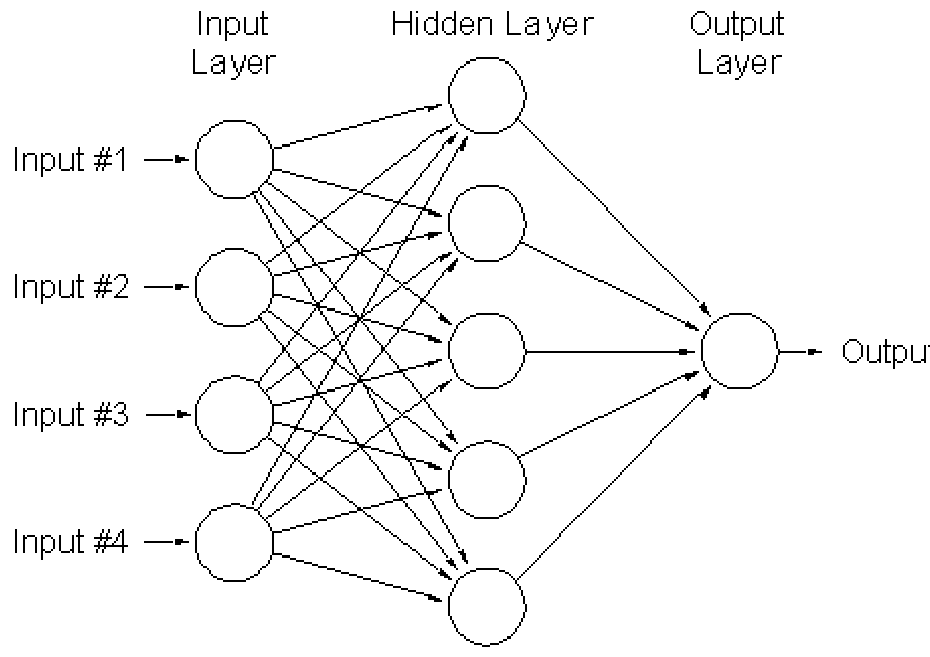
max(0.1x, x)



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

ELU $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$

A multi-layer neural network

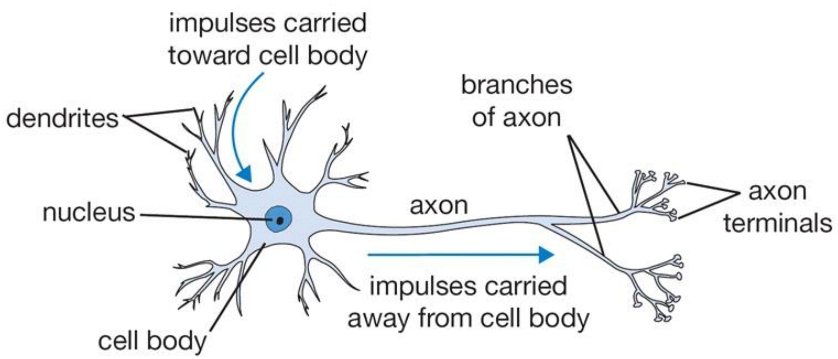


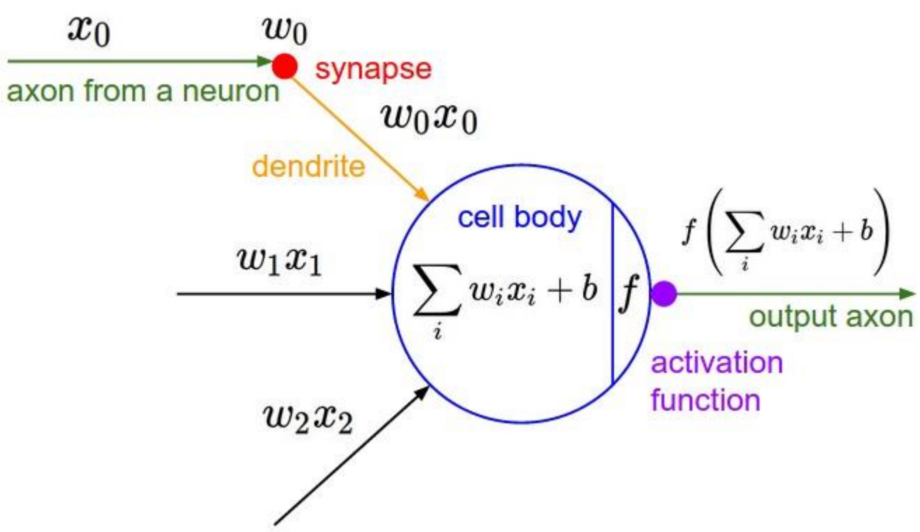
- *Nonlinear* classifier
- Can approximate any continuous function to arbitrary accuracy given sufficiently many hidden units

Lana Lazebnik

Inspiration: Neuron cells

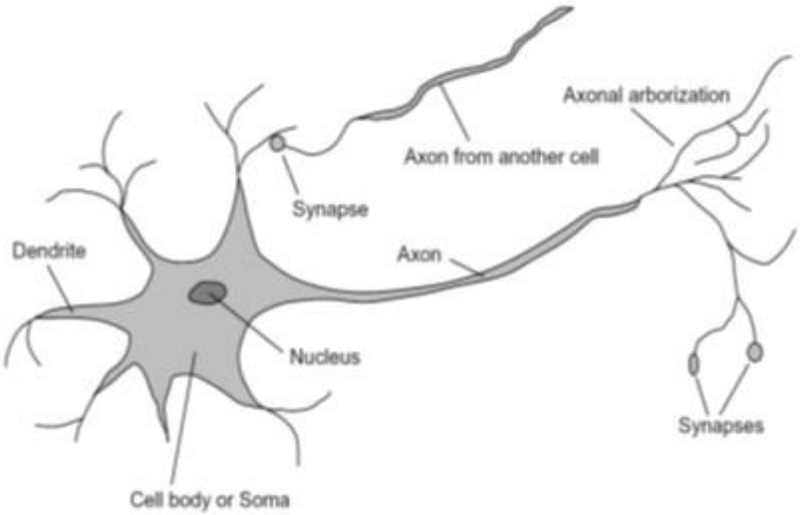
- Neurons
  - accept information from multiple inputs
  - transmit information to other neurons
- Multiply inputs by weights along edges
- Apply some function to the set of inputs at each node
- If output of function over threshold, neuron "fires"



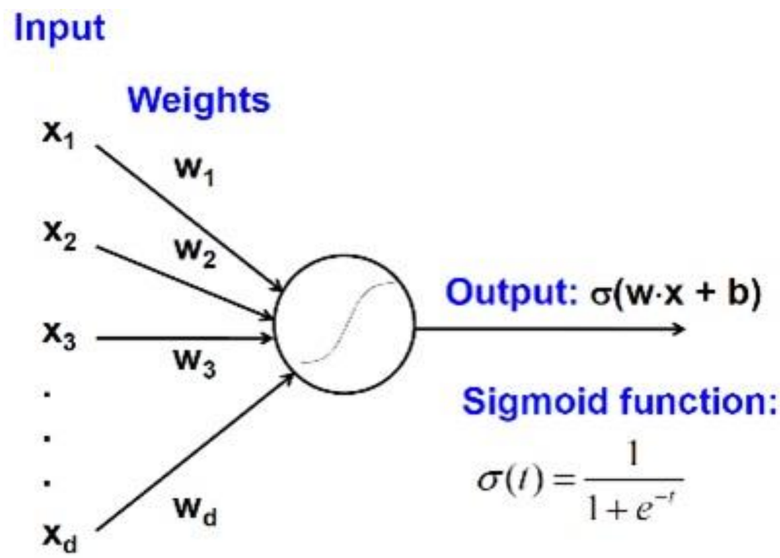


Text: HKUST, figures: Andrej Karpathy

Biological analog



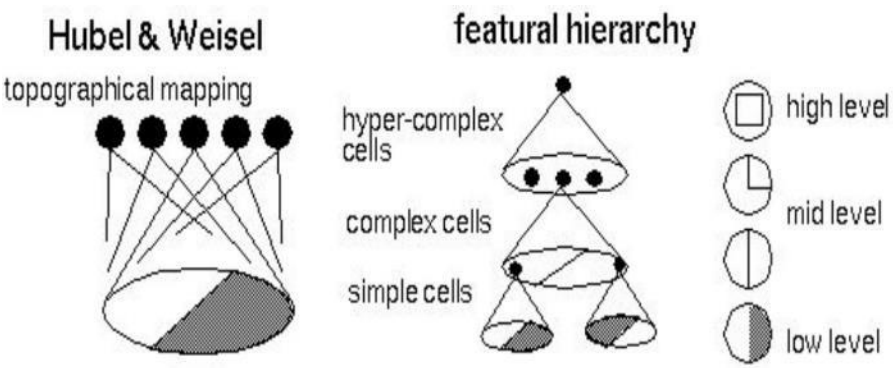
A biological neuron



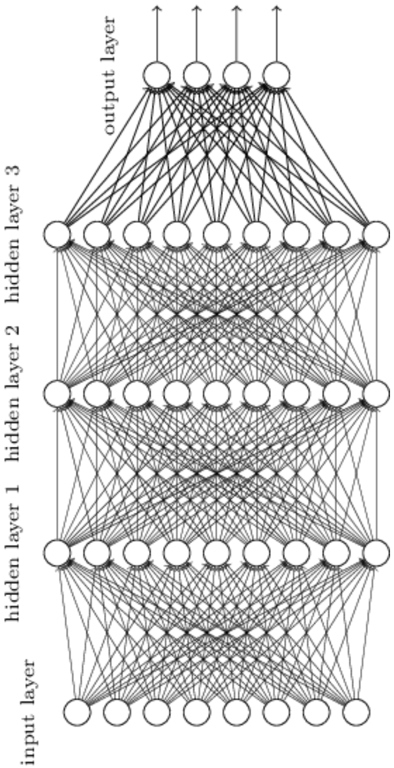
An artificial neuron

Jia-bin Huang

Biological analog



Hubel and Weisel's architecture

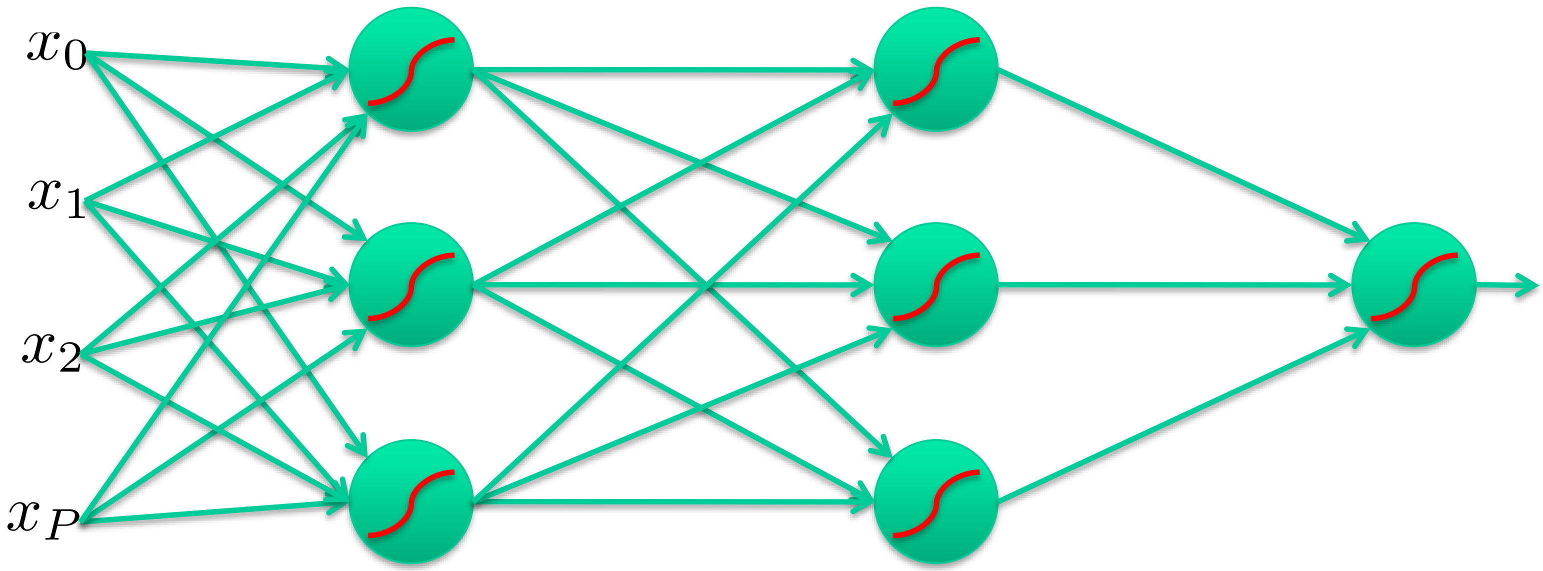


Multi-layer neural network

Adapted from Jia-bin Huang

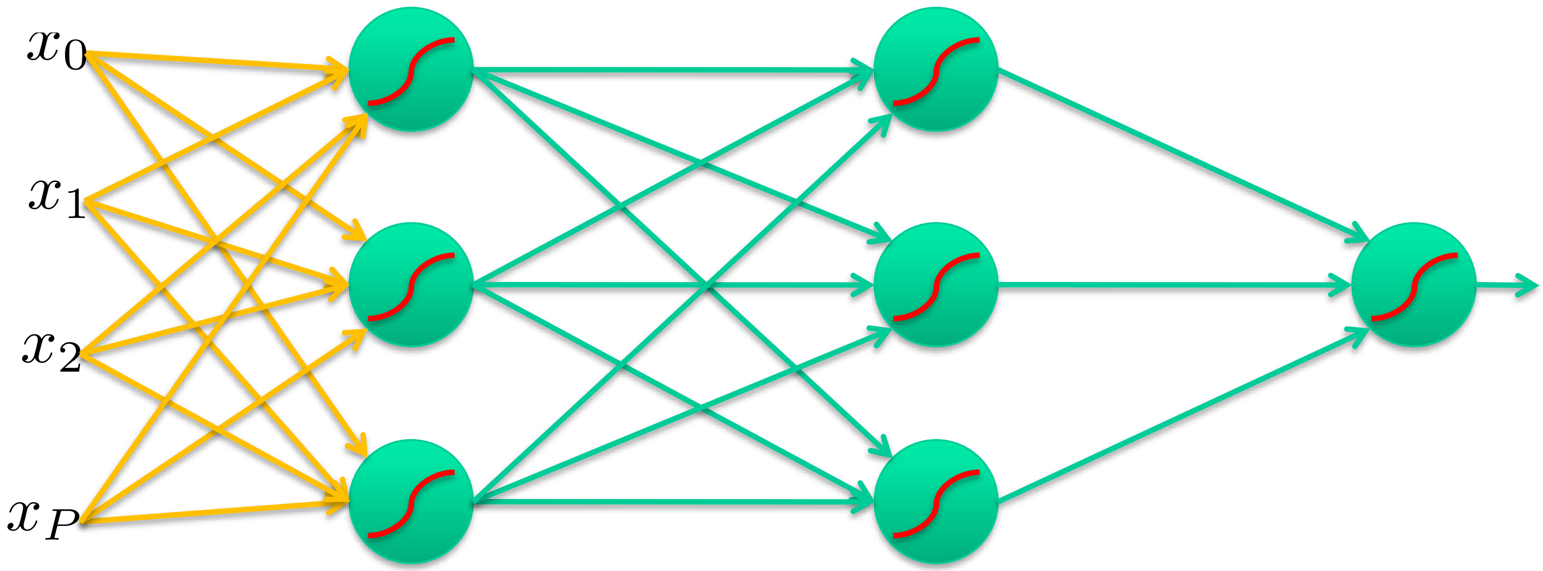
Multilayer networks

- Cascade neurons together
- Output from one layer is the input to the next
- Each layer has its own sets of weights



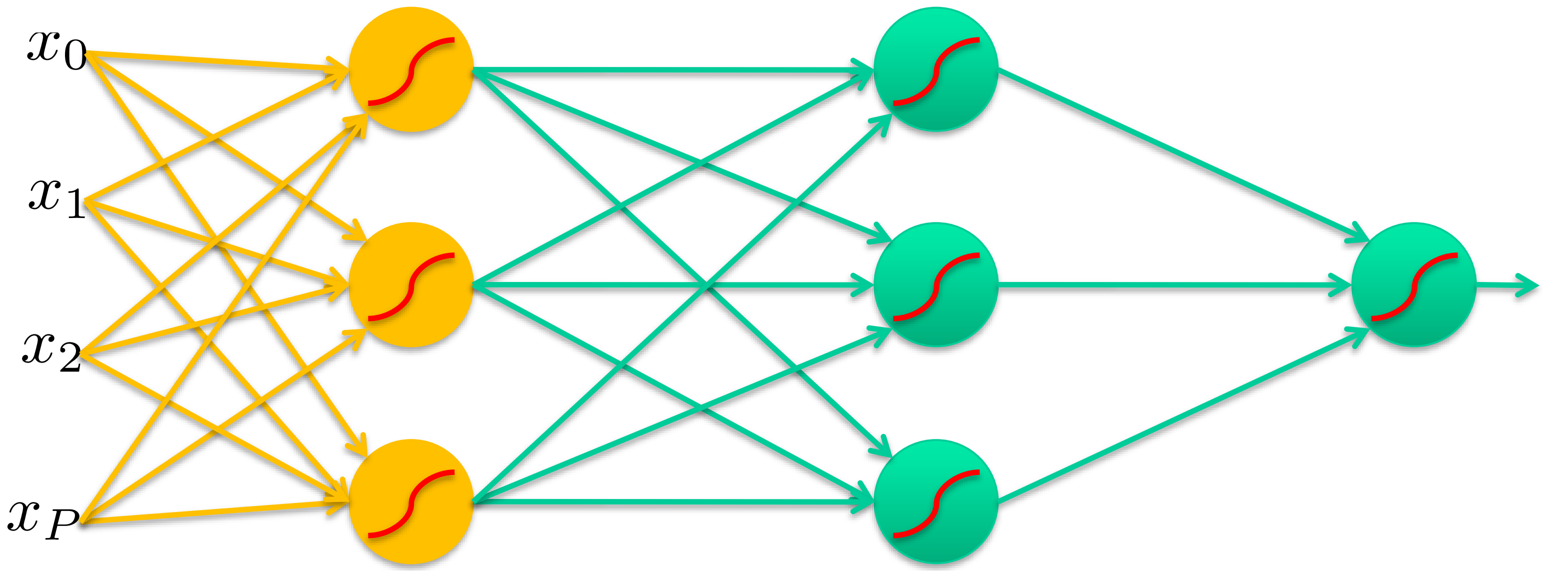
Feed-forward networks

- Predictions are fed forward through the network to classify



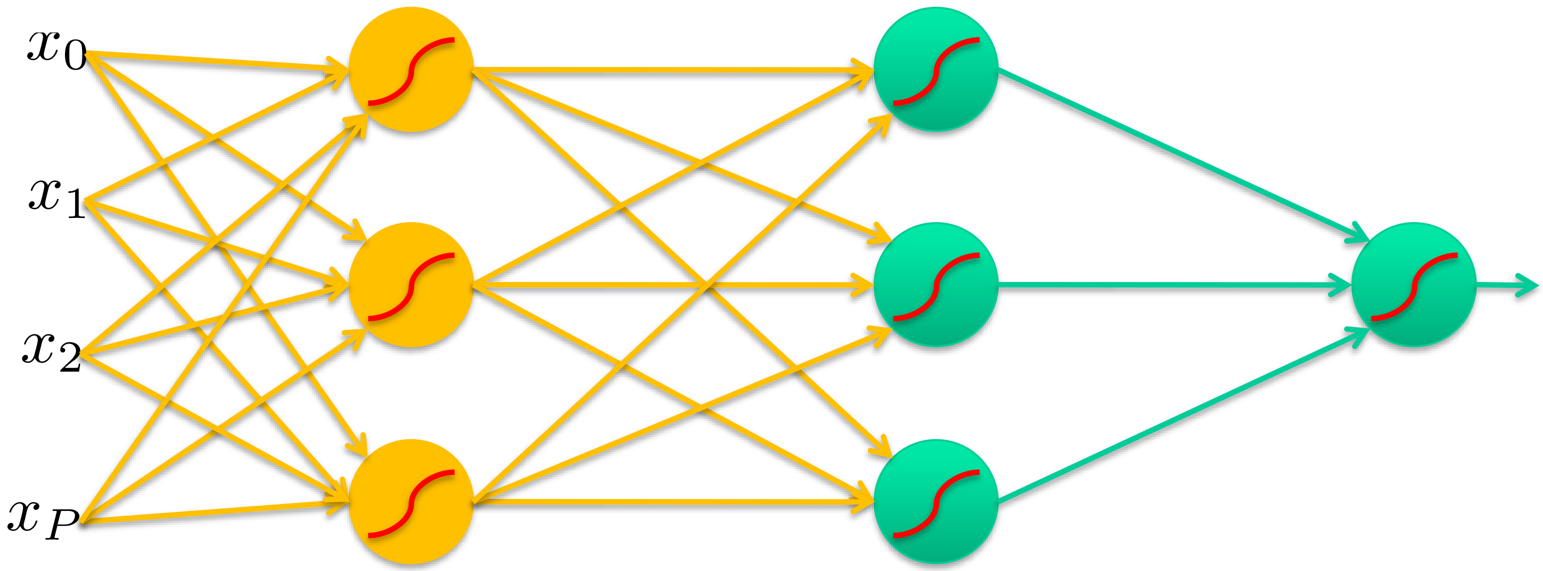
Feed-forward networks

- Predictions are fed forward through the network to classify



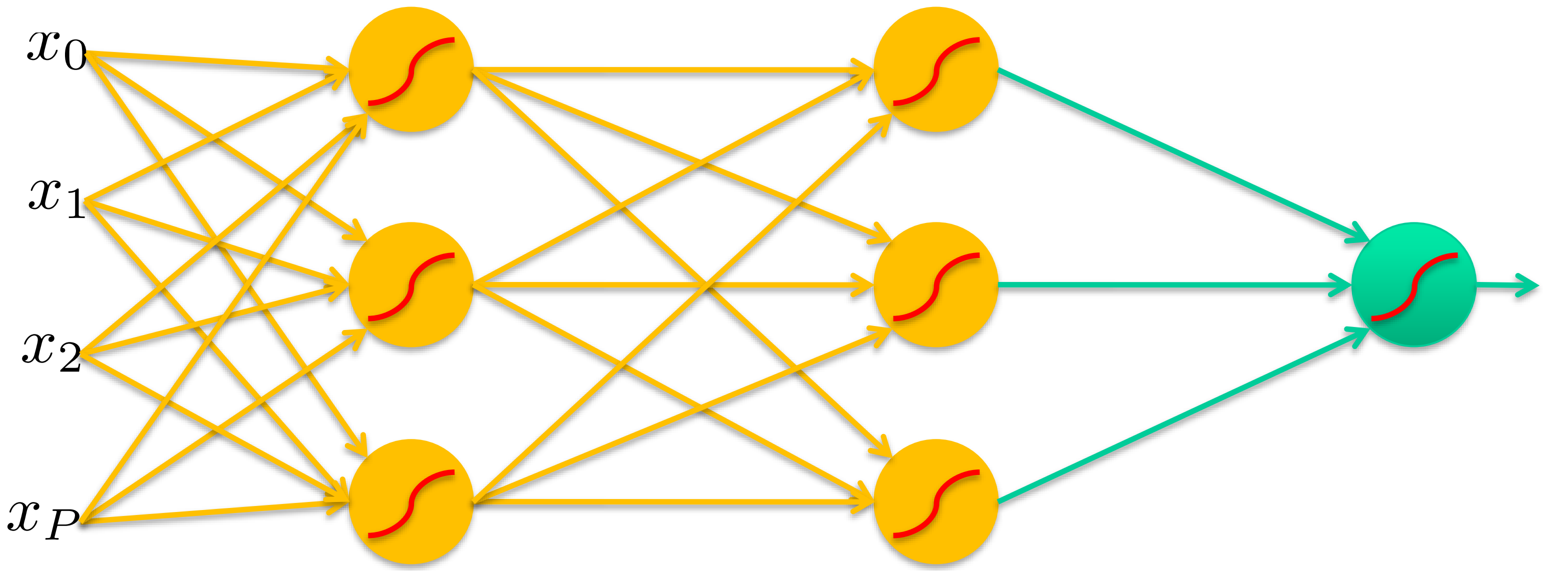
Feed-forward networks

- Predictions are fed forward through the network to classify



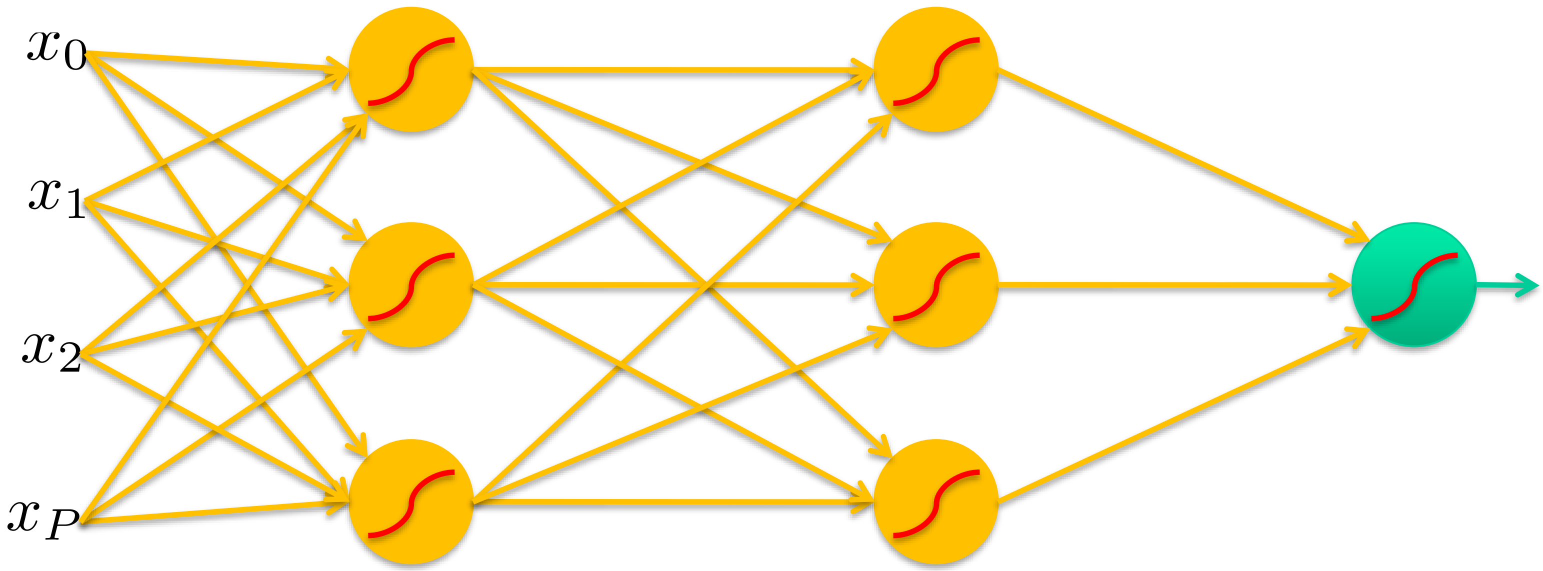
Feed-forward networks

- Predictions are fed forward through the network to classify



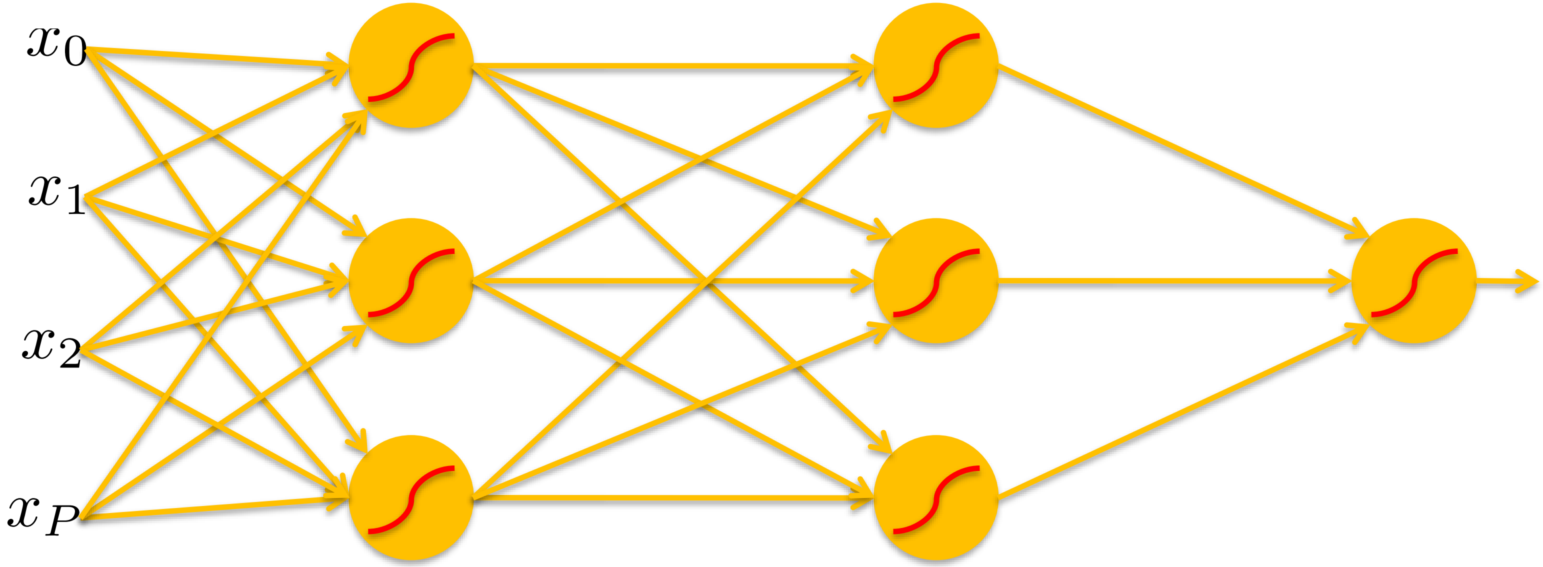
Feed-forward networks

- Predictions are fed forward through the network to classify



Feed-forward networks

- Predictions are fed forward through the network to classify



Deep neural networks

- Lots of hidden layers
- Depth = power (usually)

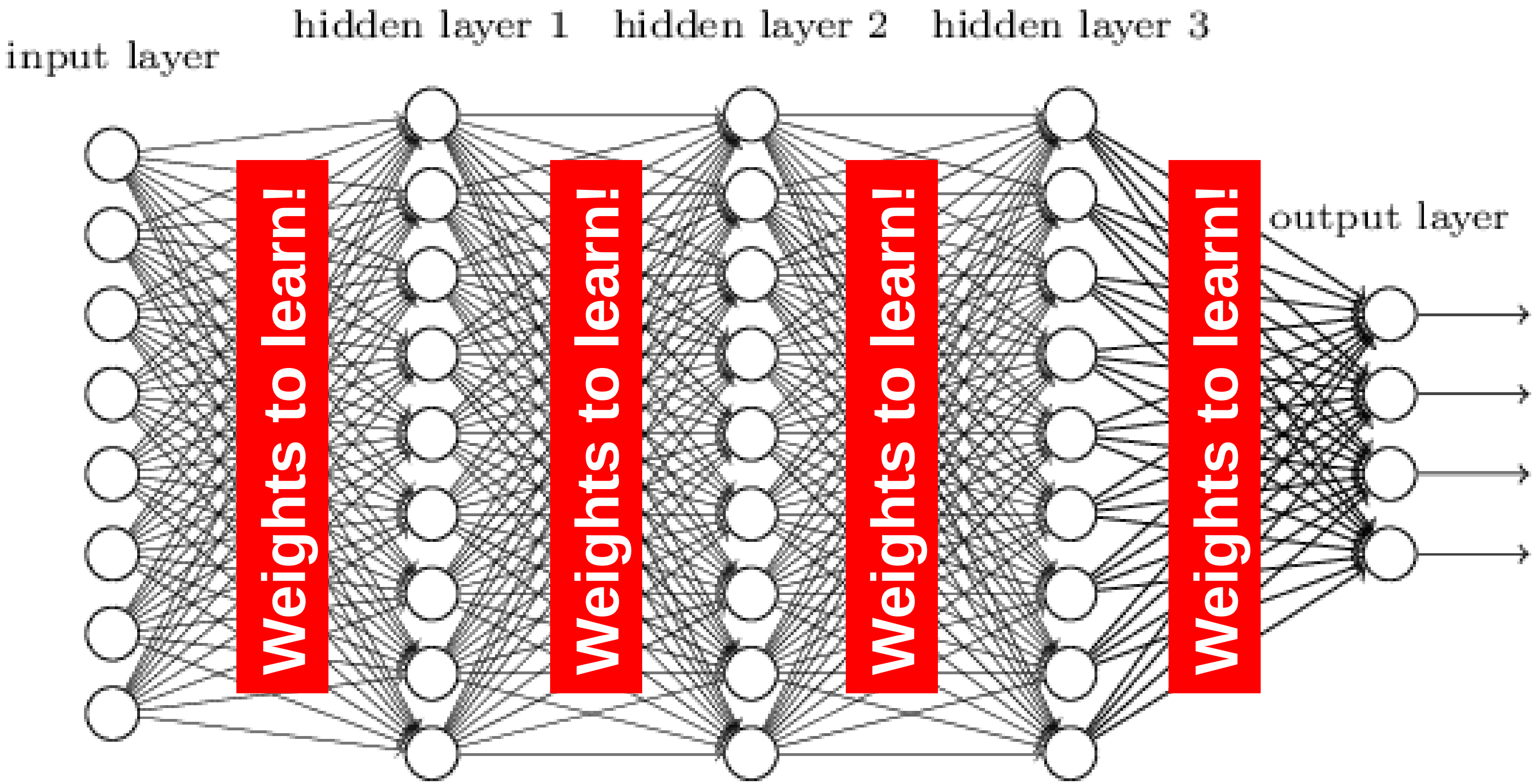
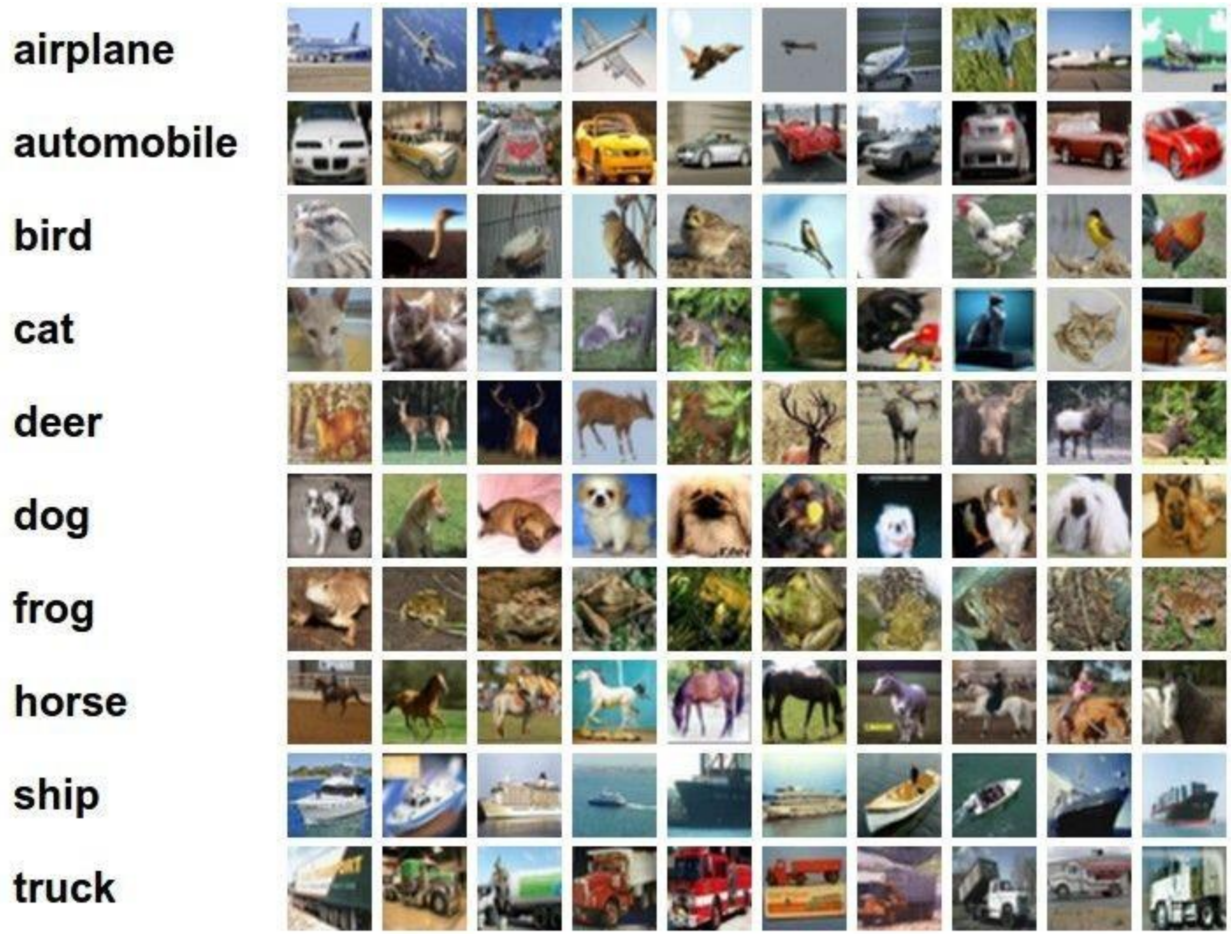


Figure from http://neuralnetworksanddeeplearning.com/chap5.html

How do we train them?

- No closed-form solution for the weights
- We will iteratively find such a set of weights that allow the outputs to match the desired outputs
- We want to minimize a loss function (a function of the weights in the network)
- For now let’s simplify and assume there’s a single layer of weights in the network

Classification goal



Example dataset: **CIFAR-10**
10 labels
50,000 training images
  each image is **32x32x3**
10,000 test images.

Classification scores

$$f(x, W) = Wx$$

f(**x**,**W**)

10 numbers, indicating class scores

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Linear classifier

$$f(x, W) = Wx \quad (+b)$$

3072x1

10x1

10x1

10x3072

[32x32x3]
array of numbers 0...1

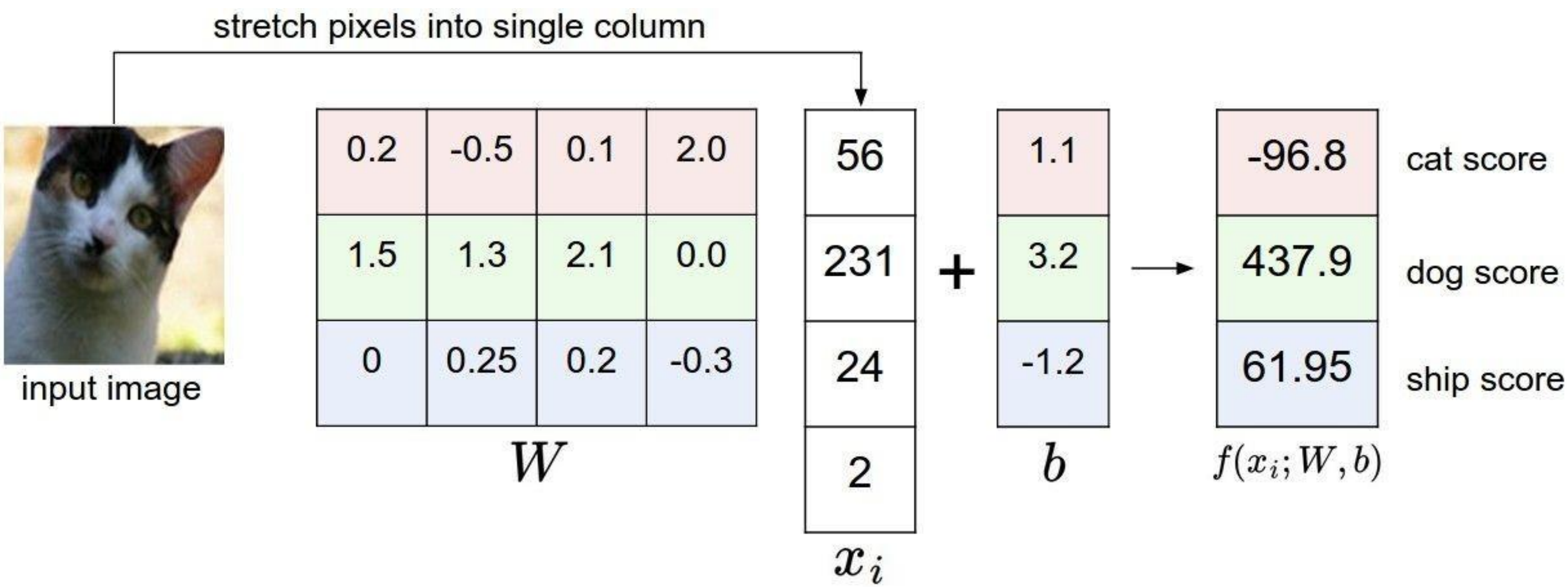
10 numbers, indicating class scores

parameters, or “weights”

Andrej Karpathy

Linear classifier

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Andrej Karpathy

Linear classifier

Going forward: Loss function/Optimization

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

2. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

Adapted from Andrej Karpathy

Linear classifier

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

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

Want: $s_{y_i} >= s_j + 1$
i.e. $s_j - s_{y_i} + 1 <= 0$

If true, loss is 0
If false, loss is magnitude of violation

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9		

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

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

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

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

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

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

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 5.3 + 1)
+ max(0, 5.6 + 1)
= 6.3 + 6.6
= 12.9

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Hinge loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the
scores vector: $s = f(x_i, W)$

the loss has the form:

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

and the full training loss is the mean
over all examples in the training data:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= 15.8 / 3 = **5.3**

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Adapted from Andrej Karpathy

Linear classifier: Hinge loss

Weight Regularization

λ = regularization strength (hyperparameter)

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \lambda R(W)$$

In common use:

L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Dropout (will see later)

Another loss: Softmax (cross-entropy)

cat	**3.2**
car	5.1
frog	-1.7

scores = unnormalized log probabilities of the classes.

$$P(Y=k|X=x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ where $$s = f(x_i; W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y=y_i|X=x_i)$$

Another loss: Softmax (cross-entropy)

$$L_i = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$$

unnormalized probabilities

	unnormalized log probabilities		unnormalized probabilities		probabilities
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ L_i = -log(0.13)
= **0.89**

unnormalized log probabilities

probabilities

Adapted from Andrej Karpathy

Other losses

- Triplet loss (Schroff, FaceNet, CVPR 2015)

$$\sum_{i}^{N}\left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]_+$$

a denotes anchor
p denotes positive
n denotes negative

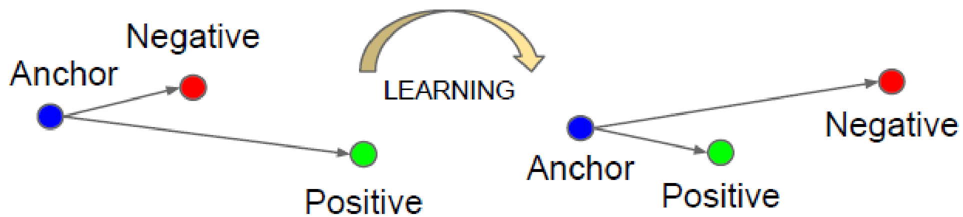


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

- Anything you want! (almost)

To minimize loss, use gradient descent

Gradient descent in multi-layer nets

- **How to update the weights at all layers?**
- Answer: backpropagation of error from higher layers to lower layers

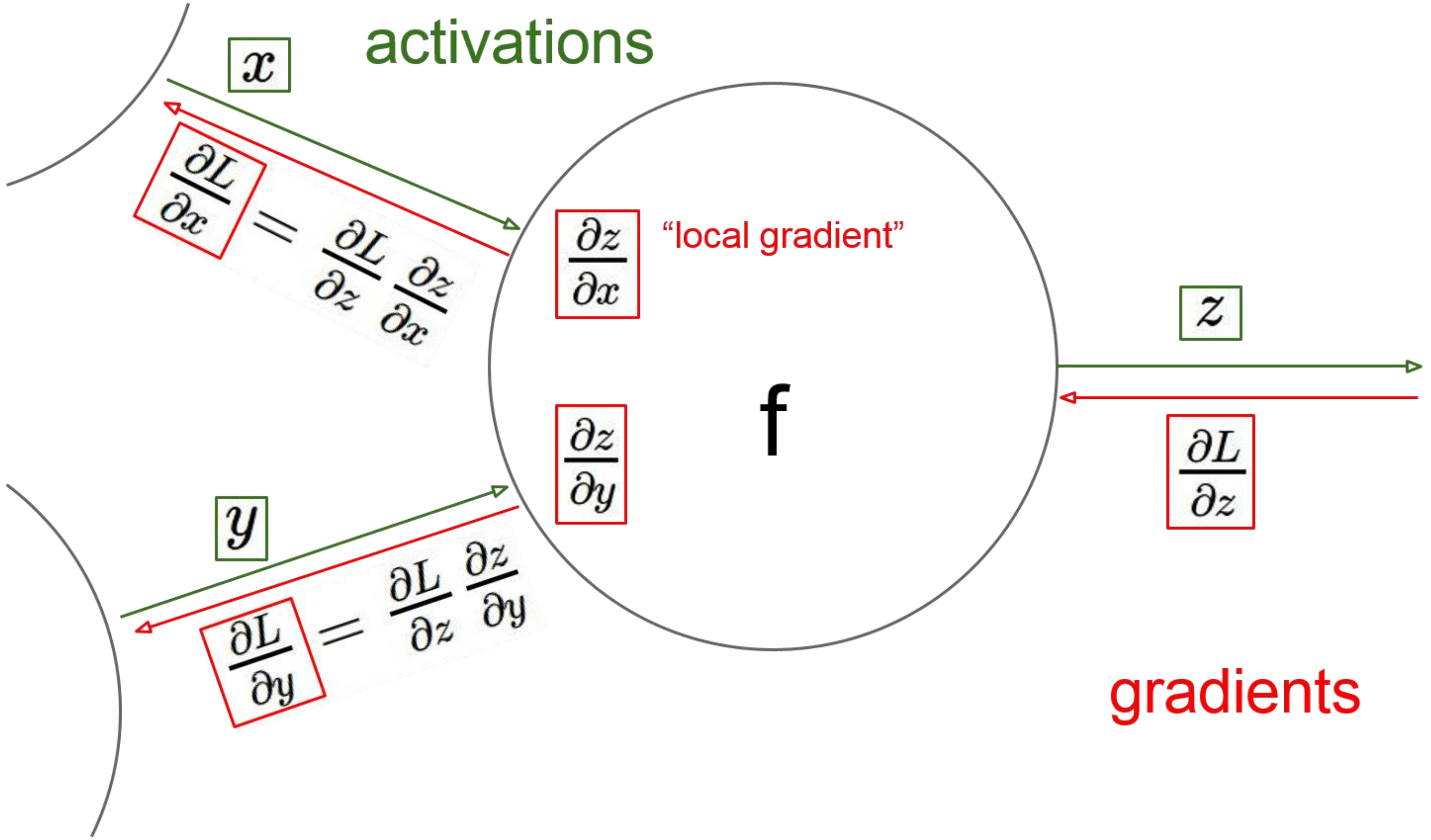


Figure from Andrej Karpathy

Computing gradient for each weight

- We need to move weights in direction opposite to gradient of loss wrt that weight:

  $w_{ji} = w_{ji} - \eta\ dE/dw_{ji}$

  $w_{kj} = w_{kj} - \eta\ dE/dw_{kj}$

- Loss depends on weights in an indirect way, so we'll use the chain rule and compute:

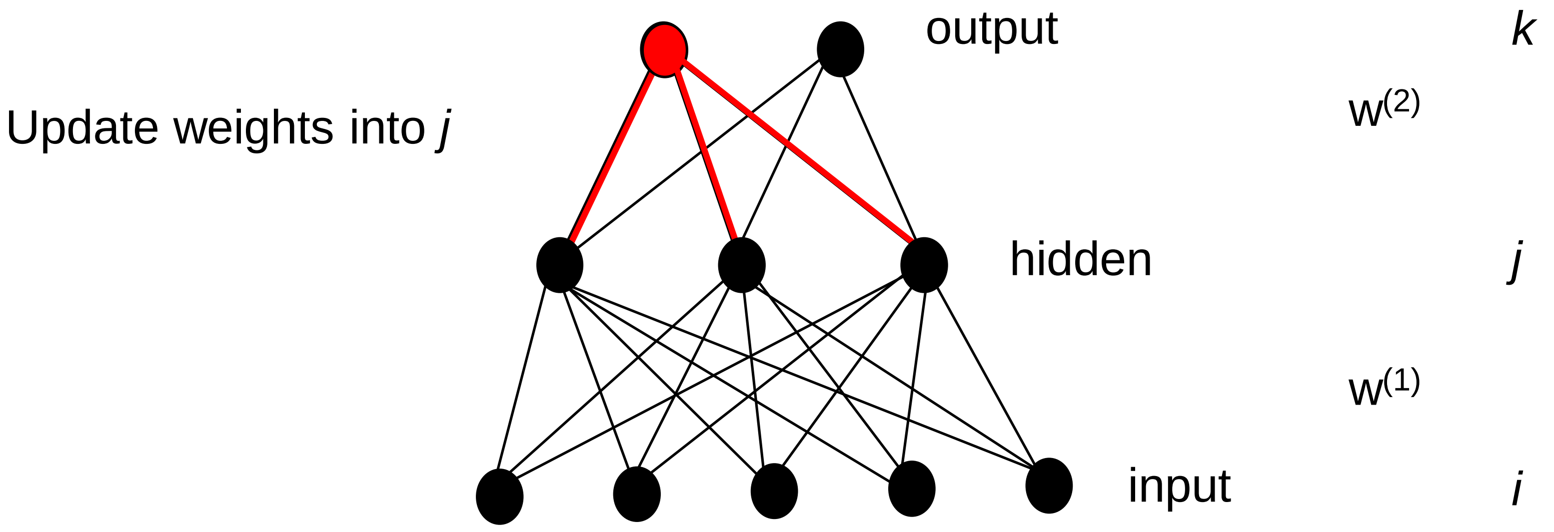
  $dE/dw_{ji} = dE/dz_j\ dz_j/da_j\ da_j/dw_{ji}$

  (and similarly for dE/dw_{kj})

- The error (dE/dz_j) is hard to compute (indirect, need chain rule again)
- We'll simplify the computation by doing it step by step via *backpropagation* of error

Backpropagation: Graphic example

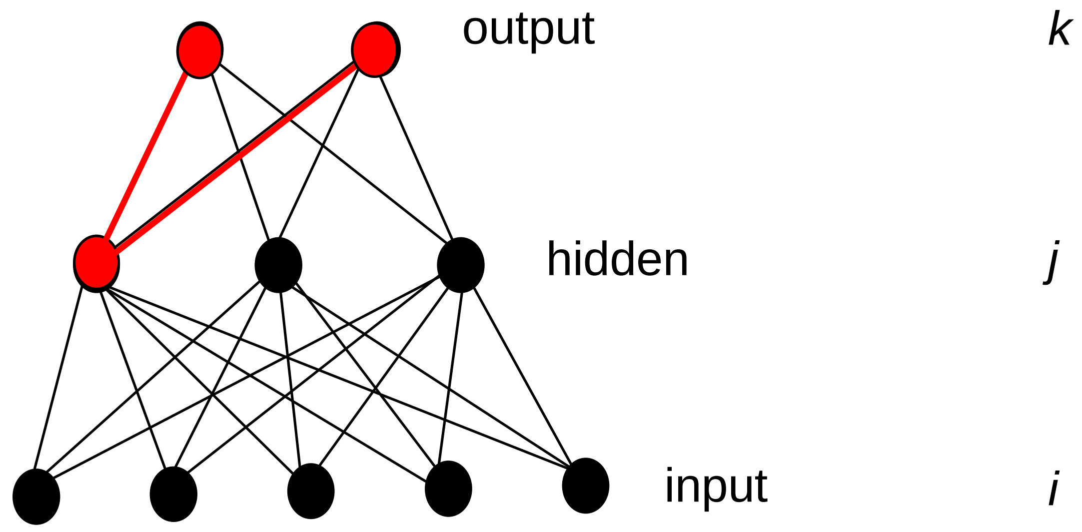
First calculate error of output units and use this to change the top layer of weights.



Adapted from Ray Mooney, equations from Chris Bishop

Backpropagation: Graphic example

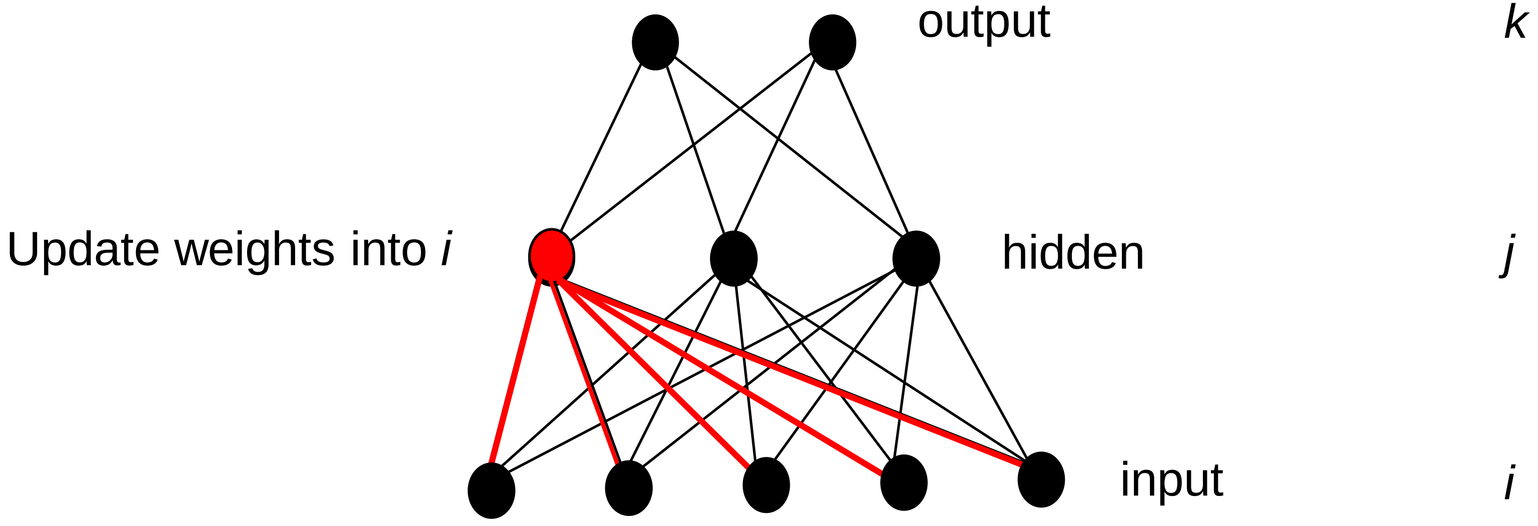
Next calculate error for hidden units based on errors on the output units it feeds into.



Adapted from Ray Mooney, equations from Chris Bishop

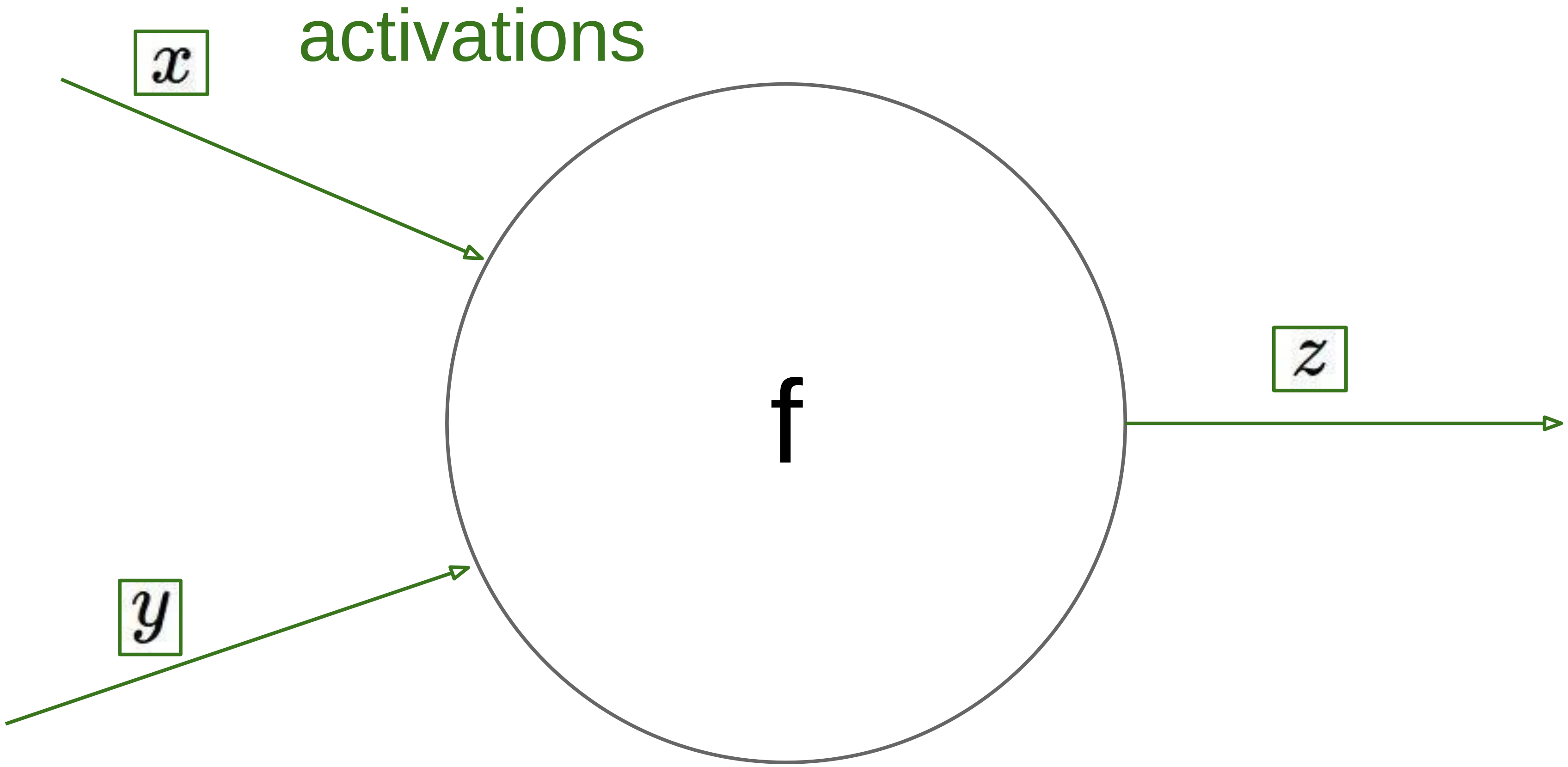
Backpropagation: Graphic example

Finally update bottom layer of weights based on errors calculated for hidden units.

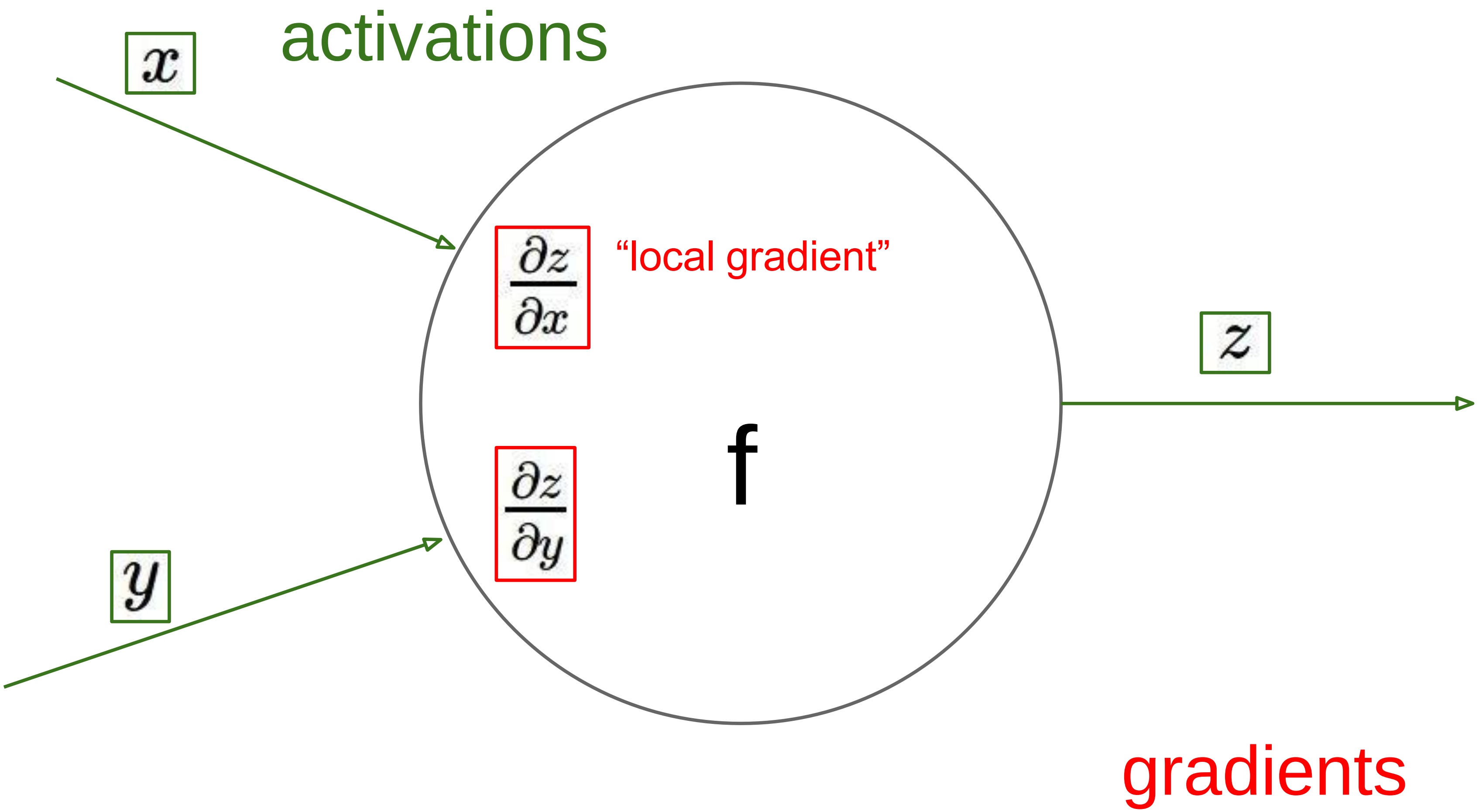


Adapted from Ray Mooney, equations from Chris Bishop

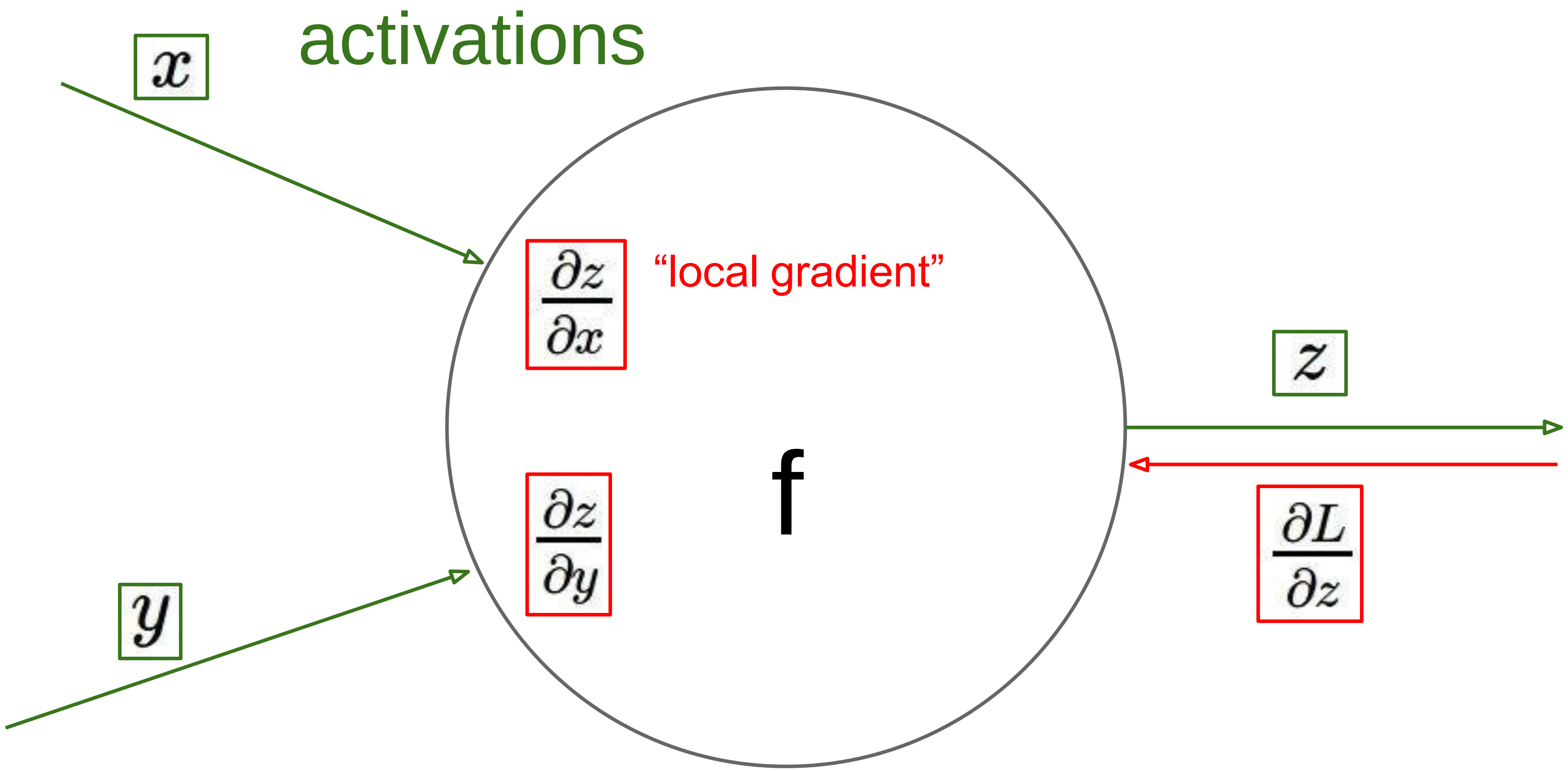
Generic example



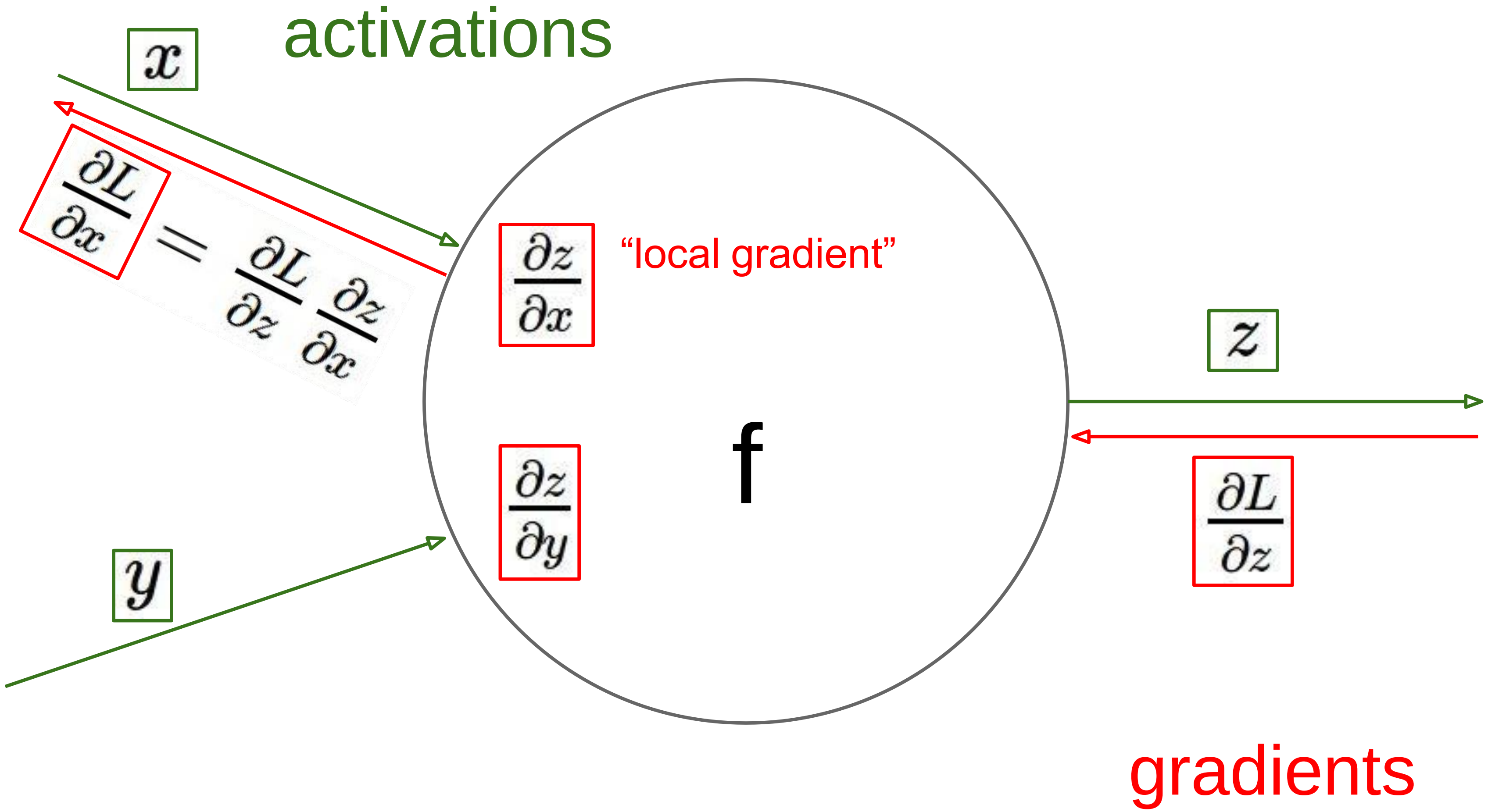
Generic example



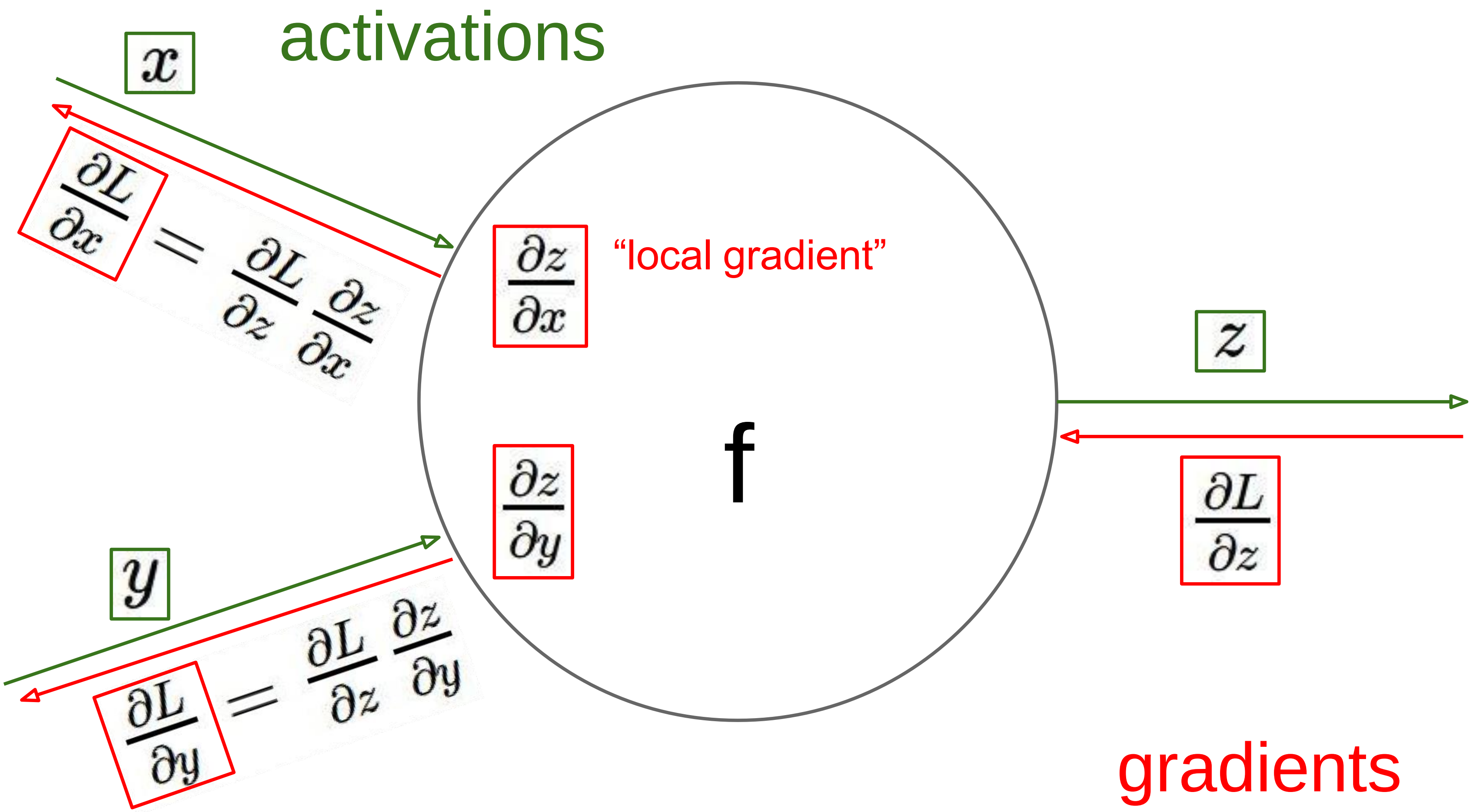
Generic example



Generic example

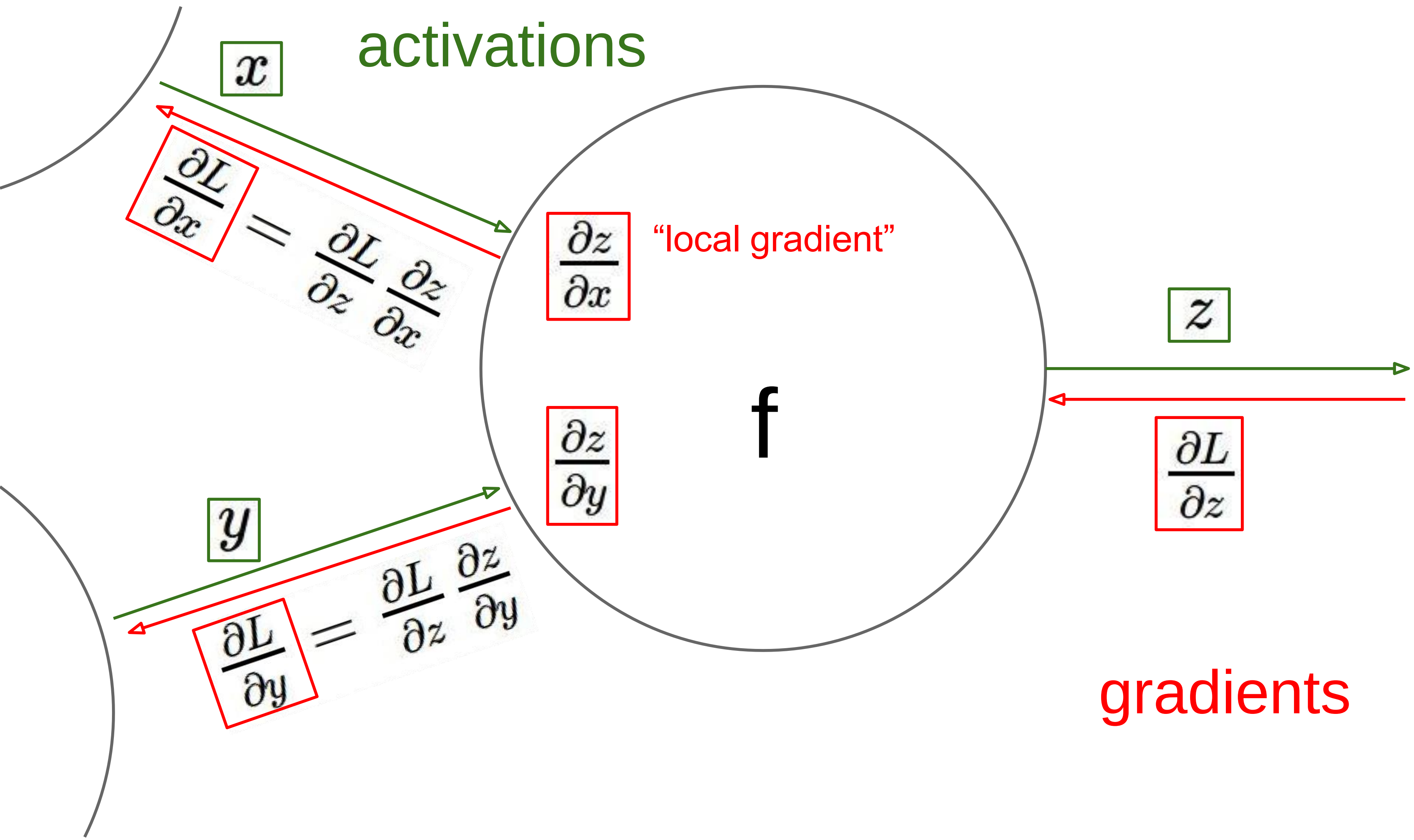


Generic example



Andrej Karpathy

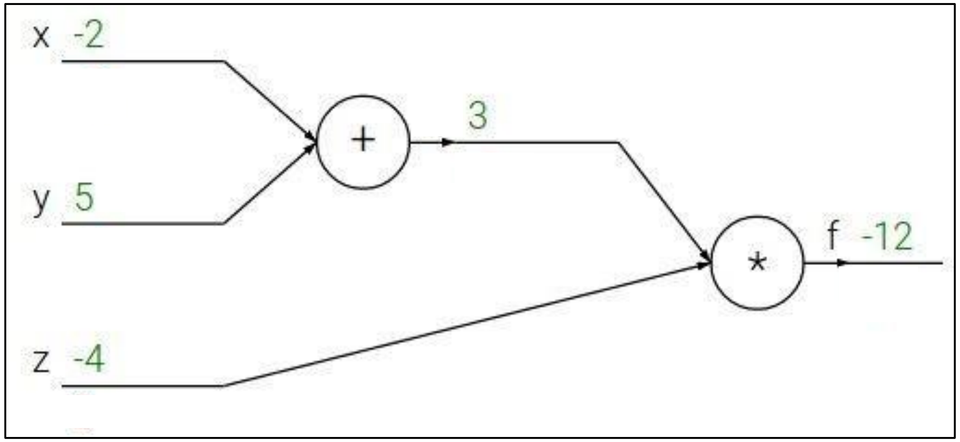
Generic example



Another generic example

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

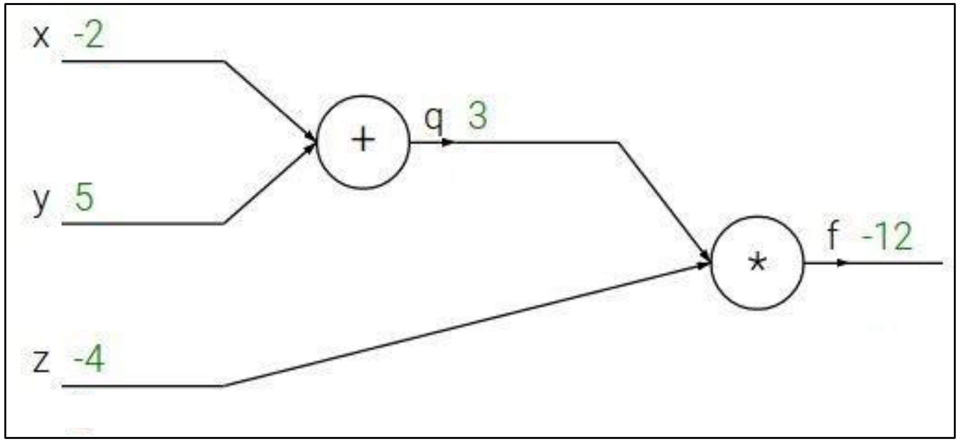
e.g. x = -2, y = 5, z = -4



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

Another generic example

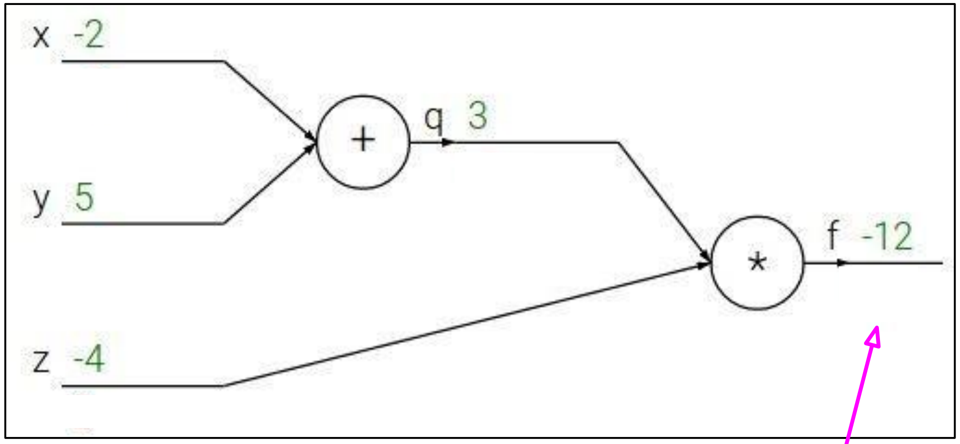
$$f(x,y,z) = (x+y)z$$

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

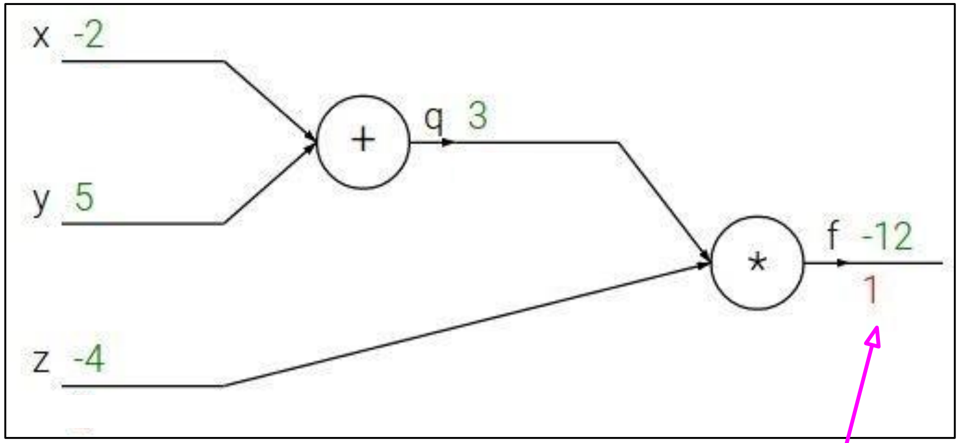


$\frac{\partial f}{\partial f}$

Another generic example

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

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



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

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

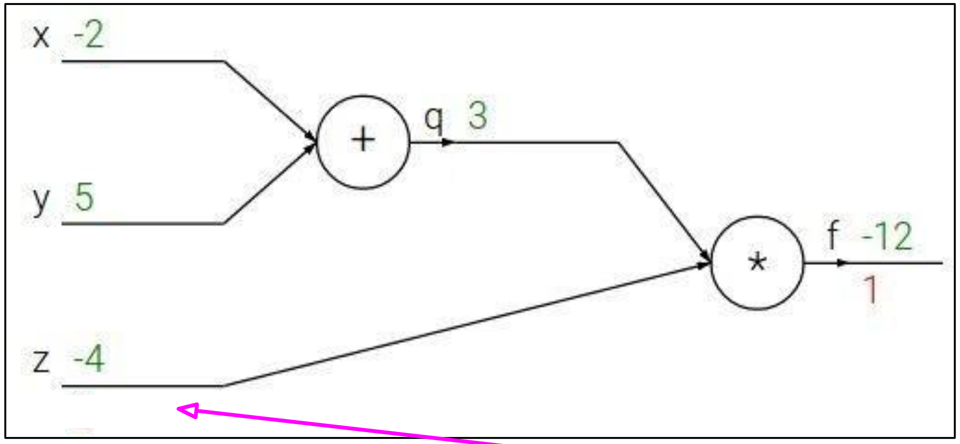
$$\frac{\partial f}{\partial f}$$

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

Another generic example

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

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



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

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

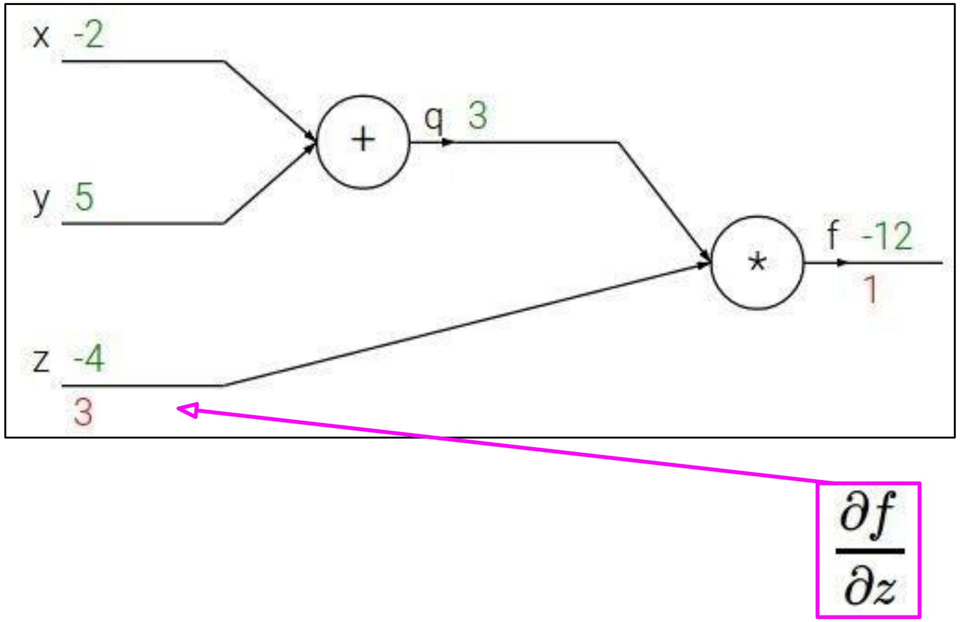
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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

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

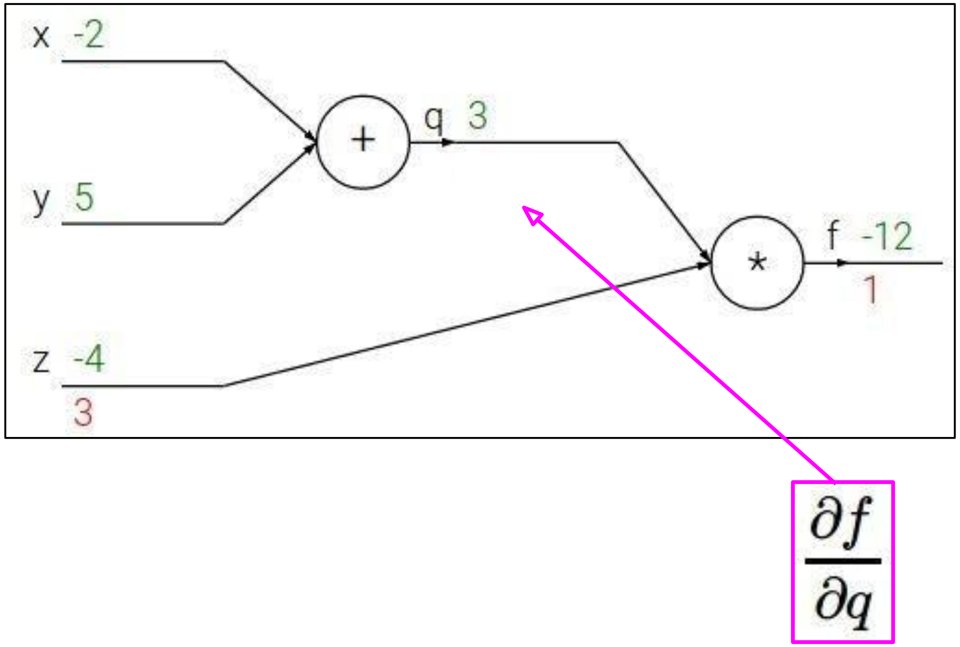
Another generic example

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

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

Another generic example

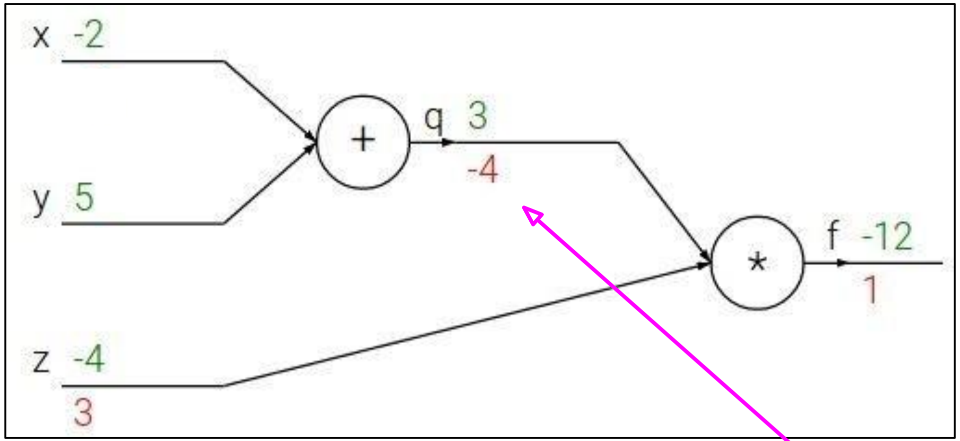
$$f(x,y,z) = (x+y)z$$

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



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

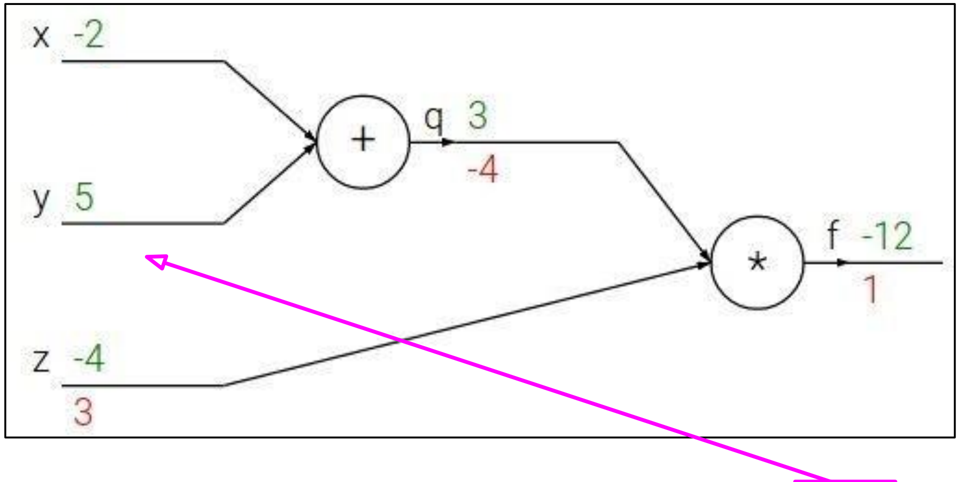
Another generic example

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

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

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

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



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

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

Another generic example

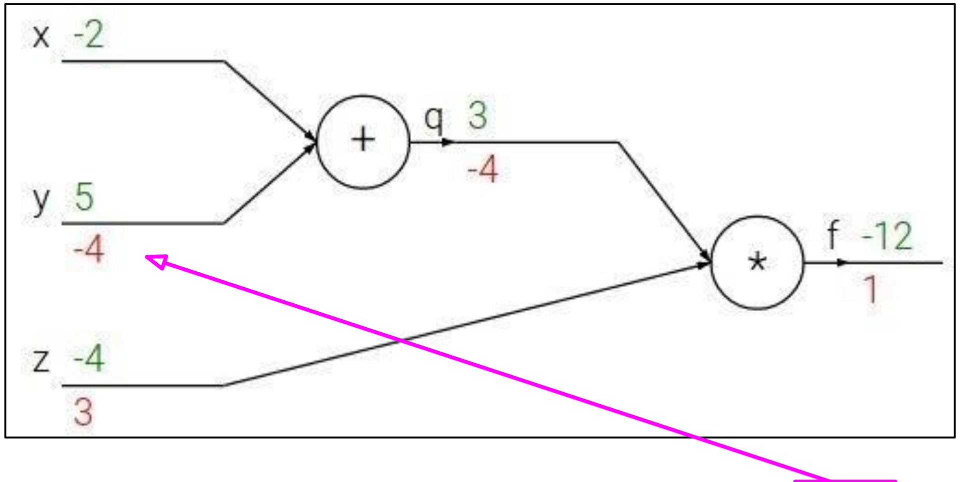
$$f(x,y,z) = (x+y)z$$

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:

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

Another generic example

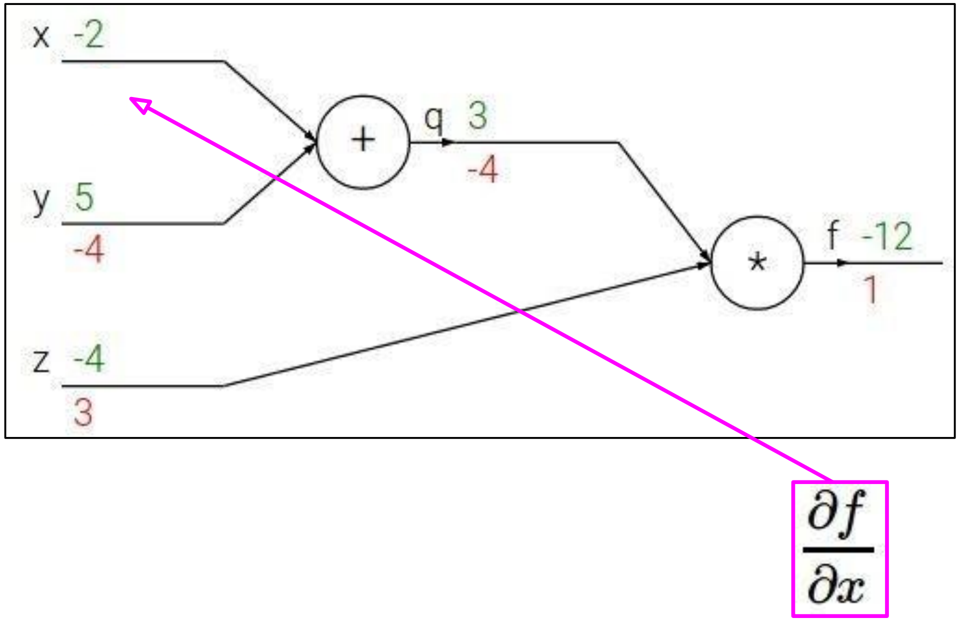
$$f(x,y,z) = (x+y)z$$

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



Another generic example

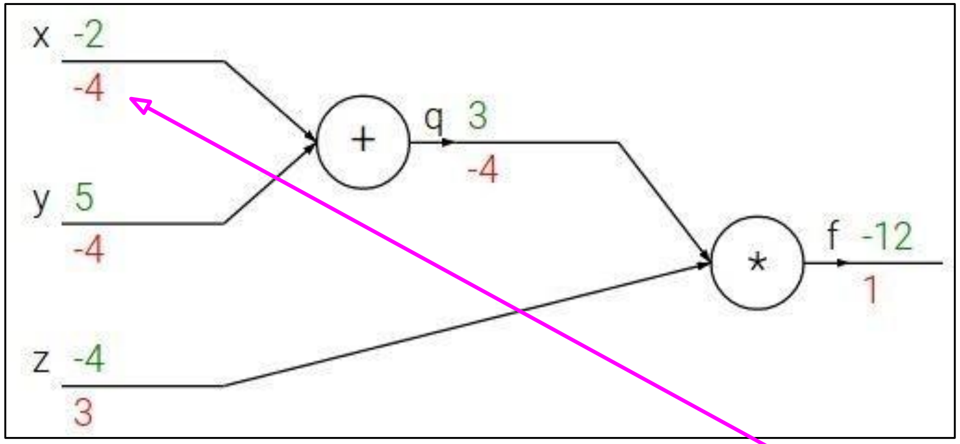
$$f(x,y,z) = (x+y)z$$

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:

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

How to compute gradient in neural net?

- In a neural network:

$$a_j = \sum_i w_{ji} z_i \qquad z_j = h(a_j)$$

- Gradient is (using chain rule):

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i$$

- Denote the “errors” as:

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j}$$

- Also:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i$$

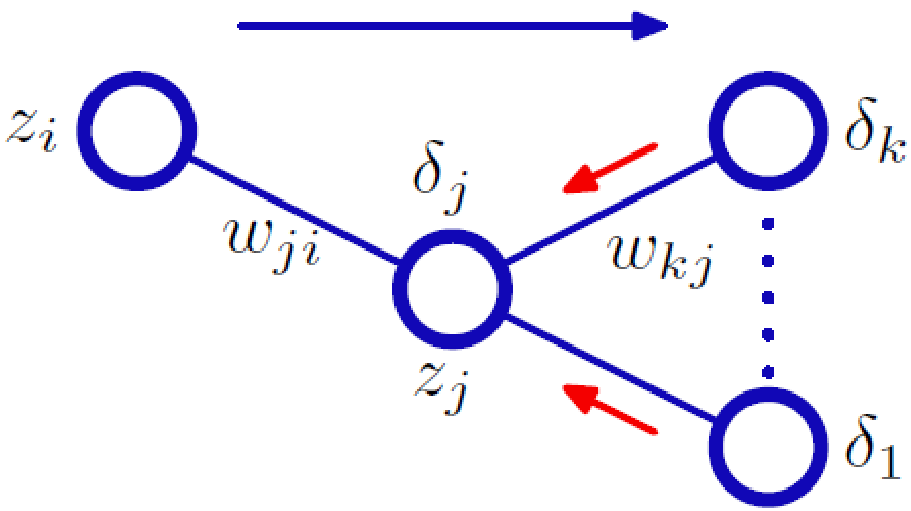
How to compute gradient in neural net?

- For output units (identity output, squared error loss): $\delta_k = y_k - t_k$
- For hidden units (using chain rule again):

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j}$$

- Backprop formula:

$$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$$



Example

- Two layer network w/ tanh at hidden layer:

$$h(a) \equiv \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

- Derivative: $h'(a) = 1 - h(a)^2$
- Minimize: $E_n = \frac{1}{2}\sum_{k=1}^{K}(y_k - t_k)^2$
- Forward propagation:

$$\begin{aligned} a_j &= \sum_{i=0}^{D} w_{ji}^{(1)} x_i \\ z_j &= \tanh(a_j) \\ y_k &= \sum_{j=0}^{M} w_{kj}^{(2)} z_j \end{aligned}$$

Example

- Errors at output (identity function at output):

$$\delta_k = y_k - t_k$$

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

- Errors at hidden units:

$$\delta_j = (1 - z_j^2) \sum_{k=1}^{K} w_{kj} \delta_k$$

- Derivatives wrt weights:

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i, \qquad \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

Same example with graphic and math

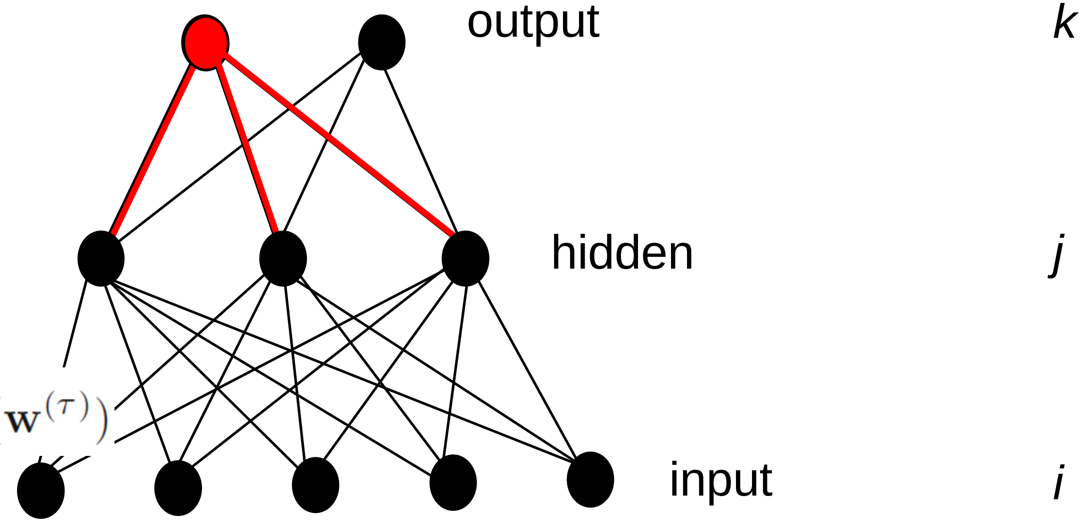
First calculate error of output units and use this to change the top layer of weights.

$$\delta_k = y_k - t_k$$

Update weights into *j*

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$



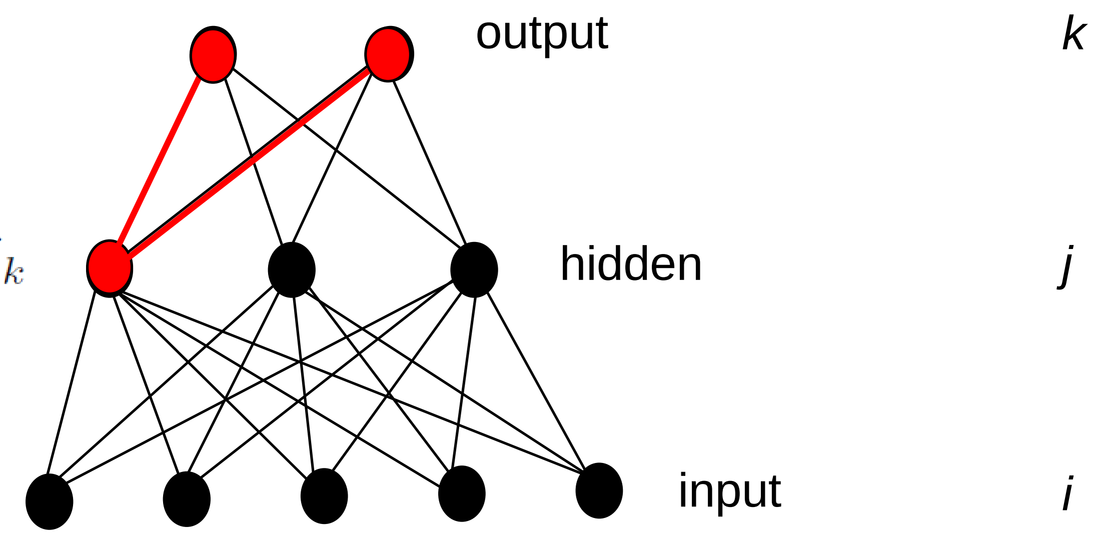
Adapted from Ray Mooney, equations from Chris Bishop

Same example with graphic and math

Next calculate error for hidden units based on errors on the output units it feeds into.

$$\delta_k = y_k - t_k$$

$$\delta_j = (1 - z_j^2) \sum_{k=1}^{K} w_{kj} \delta_k$$



Adapted from Ray Mooney, equations from Chris Bishop

Same example with graphic and math

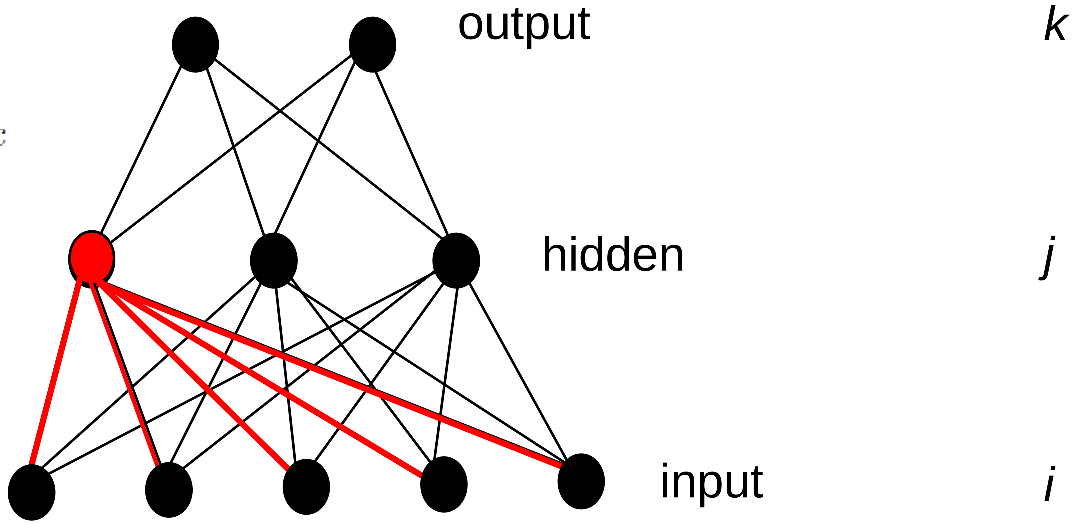
Finally update bottom layer of weights based on errors calculated for hidden units.

$$\delta_j = (1 - z_j^2) \sum_{k=1}^{K} w_{kj} \delta_k$$

Update weights into *i*

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j x_i$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$



Adapted from Ray Mooney, equations from Chris Bishop

Example: algorithm for sigmoid, sqerror

- Initialize all weights to small random values
- Until convergence (e.g. all training examples' error small, or error stops decreasing) repeat:
  - For each `(x, t=class(x))` in training set:
    - Calculate network outputs: y_k
    - Compute errors (gradients wrt activations) for each unit:
      - » $\delta_k = y_k (1-y_k)\ (y_k - t_k)$ for output units
      - » $\delta_j = z_j (1-z_j) \sum_k w_{kj}\ \delta_k$ for hidden units
    - Update weights:
      - » $w_{kj} = w_{kj} - \eta\ \delta_k\ z_j$ for output units
      - » $w_{ji} = w_{ji} - \eta\ \delta_j\ x_i$ for hidden units

Recall: $w_{ji} = w_{ji} - \eta\ dE/dz_j\ dz_j/da_j\ da_j/dw_{ji}$

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

Adapted from R. Hwa, R. Mooney

Comments on training algorithm

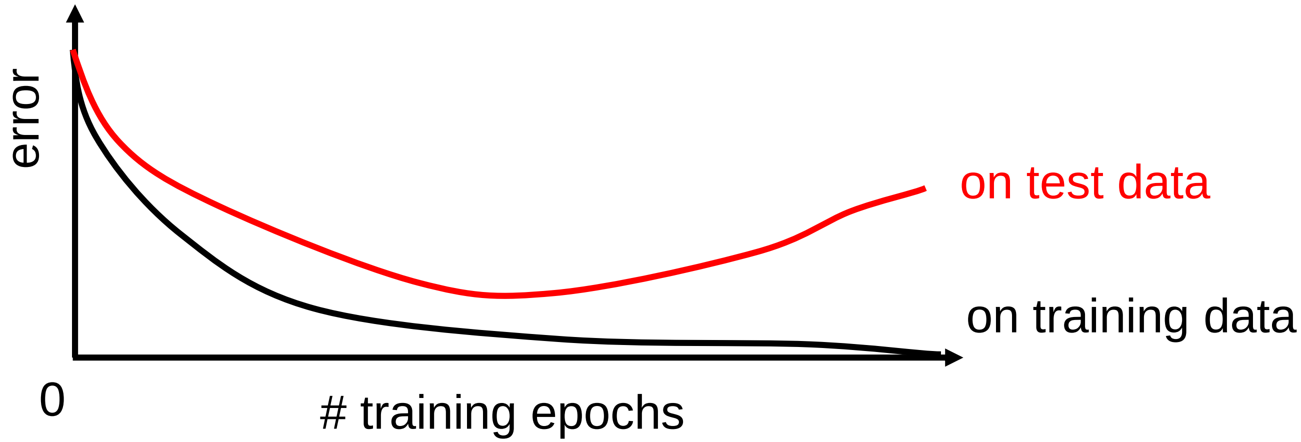
- Not guaranteed to converge to zero training error, may converge to local optima or oscillate indefinitely.
- However, in practice, does converge to low error for many large networks on real data.
- Thousands of epochs (epoch = network sees all training data once) may be required, hours or days to train.
- To avoid local-minima problems, run several trials starting with different random weights (*random restarts*), and take results of trial with lowest training set error.
- May be hard to set learning rate and to select number of hidden units and layers.
- Neural networks had fallen out of fashion in 90s, early 2000s; back with a new name and significantly improved performance (deep networks trained with dropout and lots of data).

Dealing with sparse data

- Deep neural networks require lots of data, and can overfit easily
- The more weights you need to learn, the more data you need
- That's why with a deeper network, you need more data for training than for a shallower network
- Ways to prevent overfitting include:
  - Using a validation set to stop training or pick parameters
  - Regularization
  - Transfer learning
  - Data augmentation

Over-training prevention

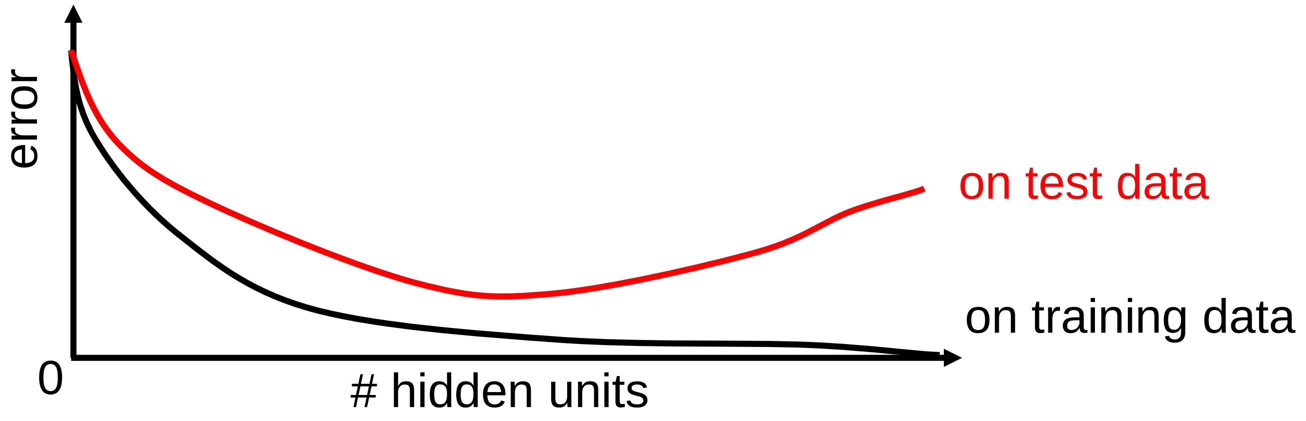
- Running too many epochs can result in over-fitting.



- Keep a hold-out validation set and test accuracy on it after every epoch. Stop training when additional epochs actually increase validation error.

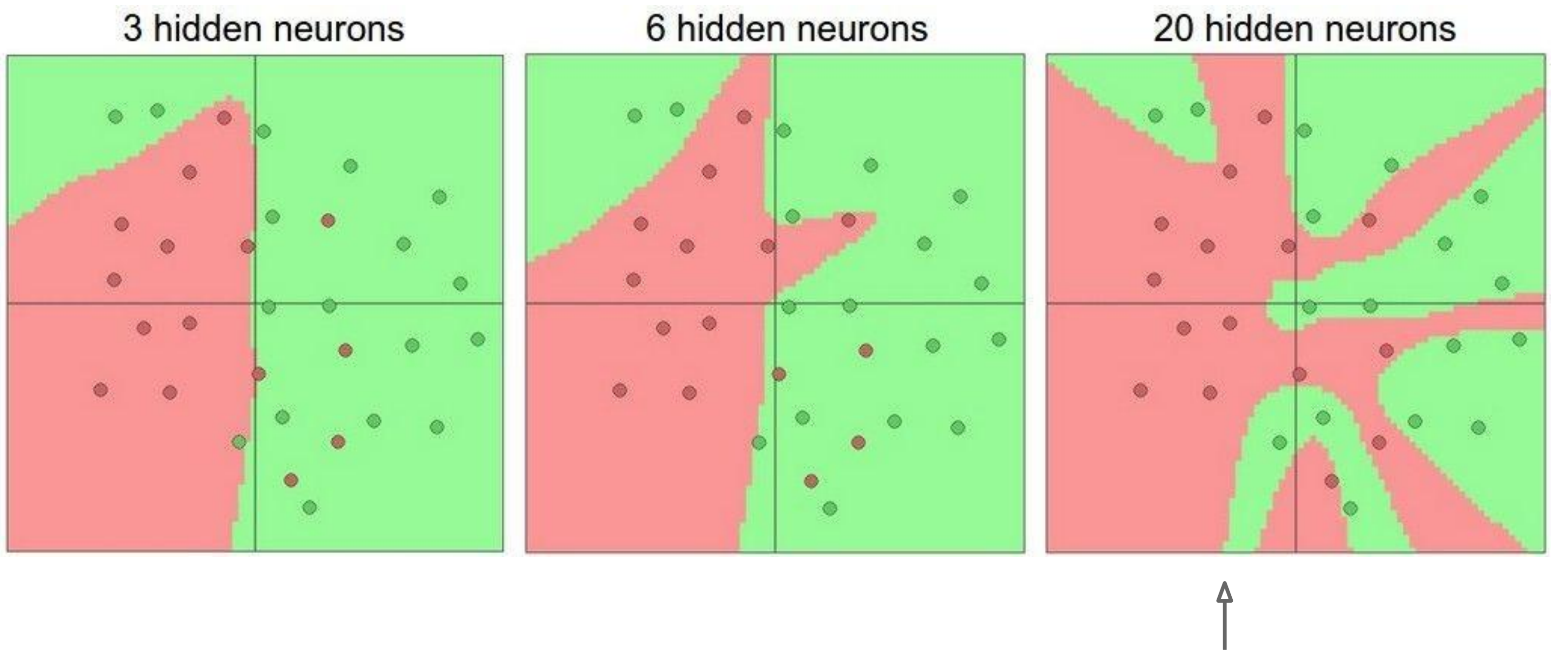
Determining best number of hidden units

- Too few hidden units prevents the network from adequately fitting the data.
- Too many hidden units can result in over-fitting.



- Use internal cross-validation to empirically determine an optimal number of hidden units.

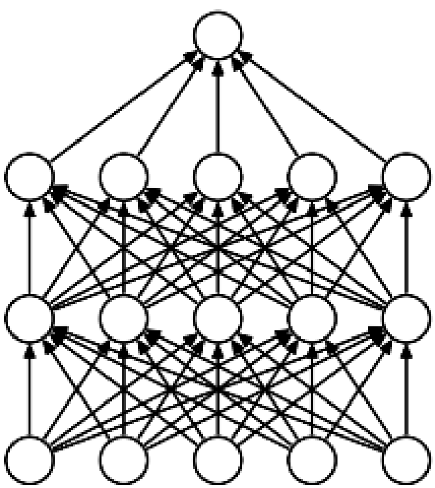
Effect of number of neurons



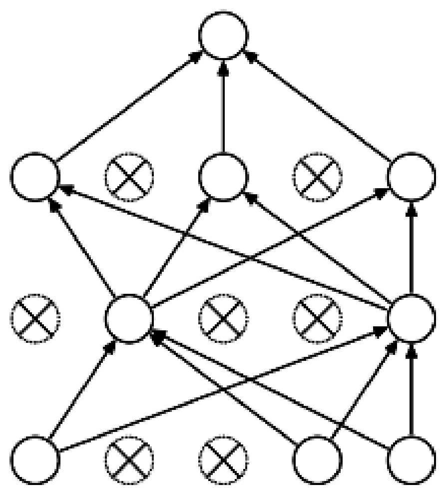
more neurons = more capacity

Regularization

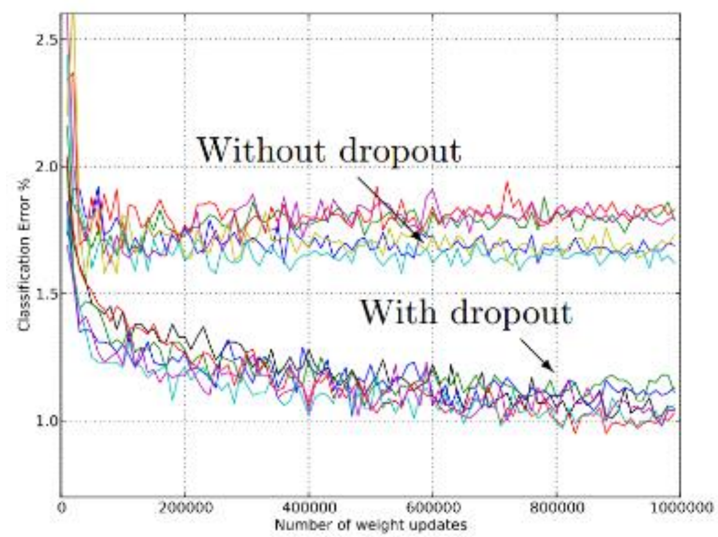
- L1, L2 regularization (*weight decay*)
- Dropout
  - Randomly turn off some neurons
  - Allows individual neurons to independently be responsible for performance



(a) Standard Neural Net



(b) After applying dropout.

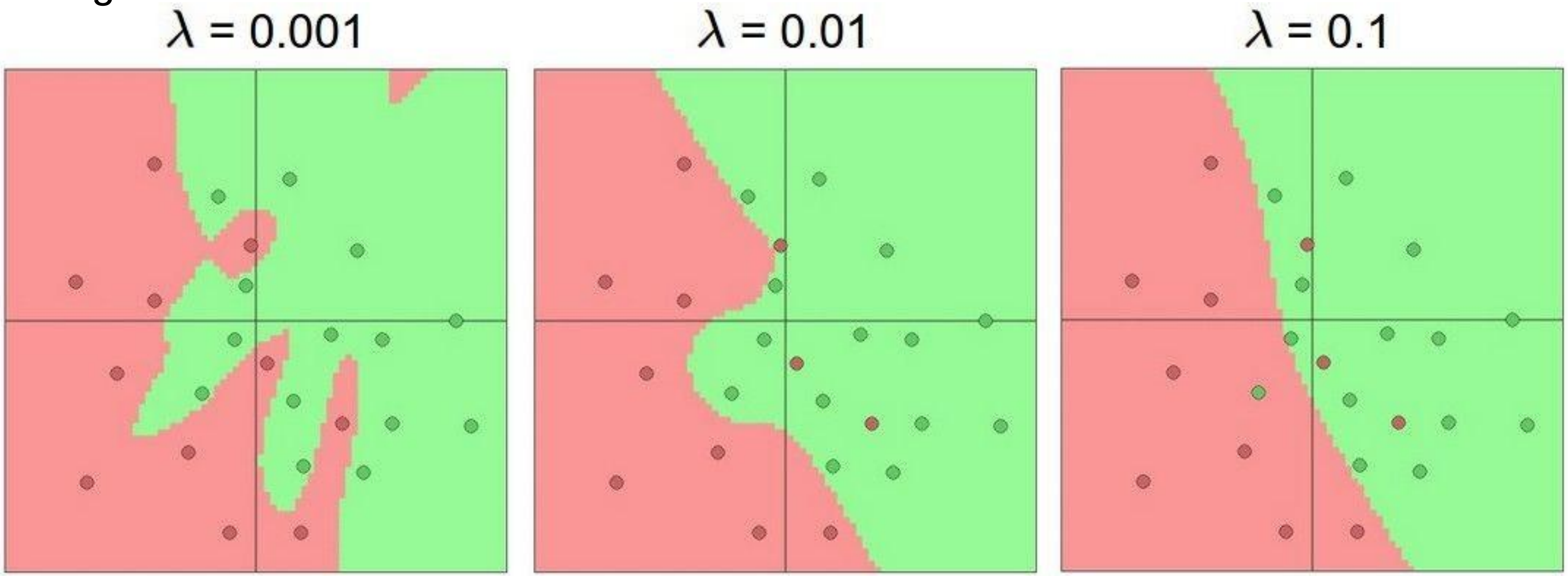


Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Adapted from Jia-bin Huang

Effect of regularization

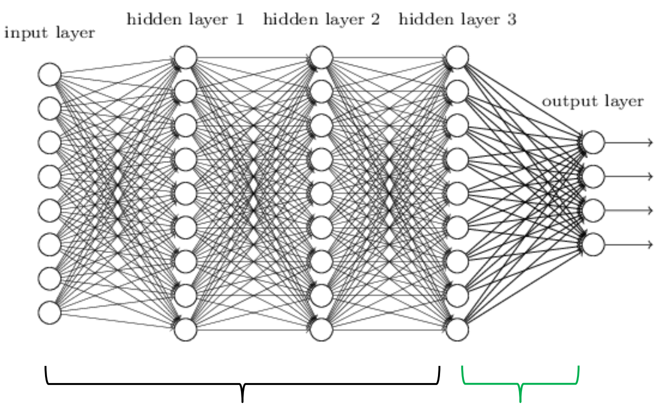
Do not use size of neural network as a regularizer. Use stronger regularization instead:



(you can play with this demo over at ConvNetJS: http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html)

Transfer learning

- If you have sparse data in your domain of interest *(target)*, but have rich data in a disjoint yet related domain *(source)*,
- You can train the early layers on the *source* domain, and only the last few layers on the *target domain:*

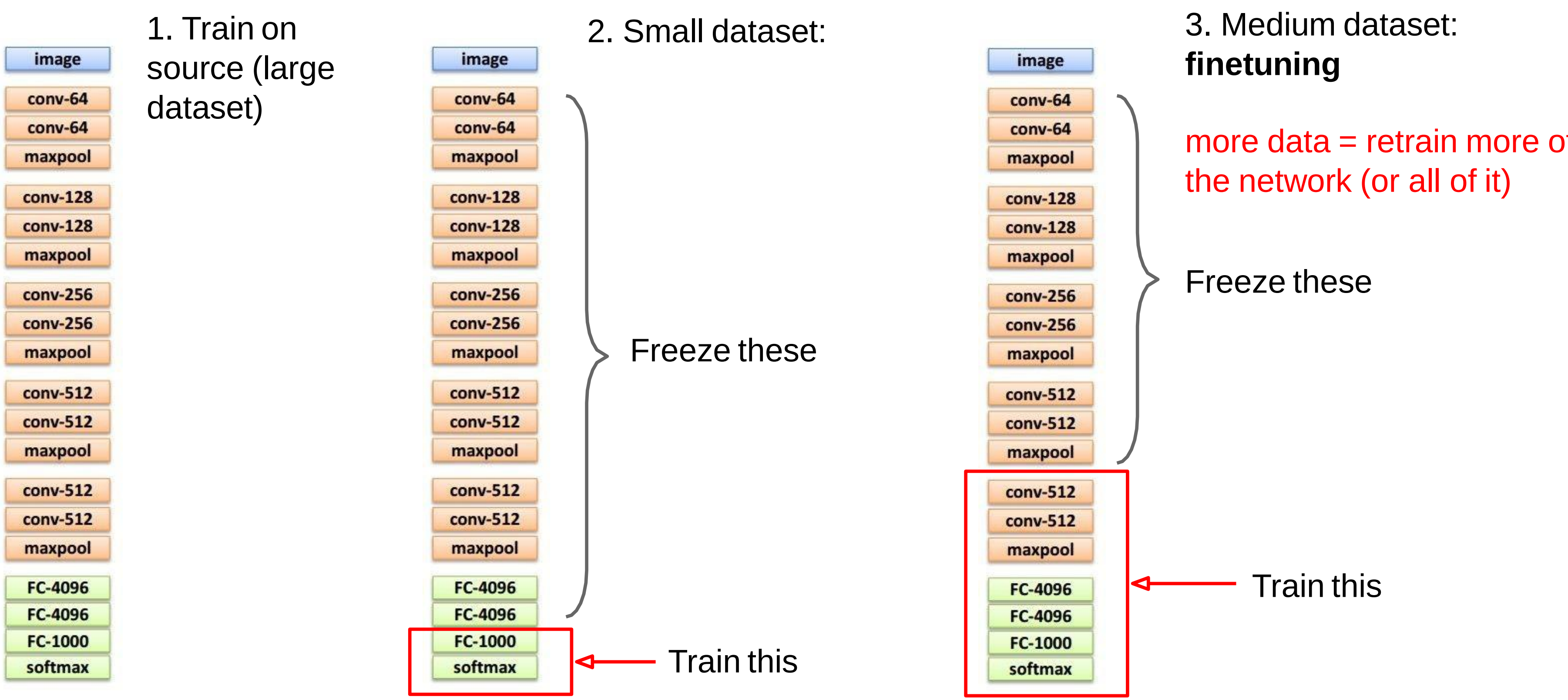


Set these to the already learned weights from another network

Learn these on your own task

Transfer learning

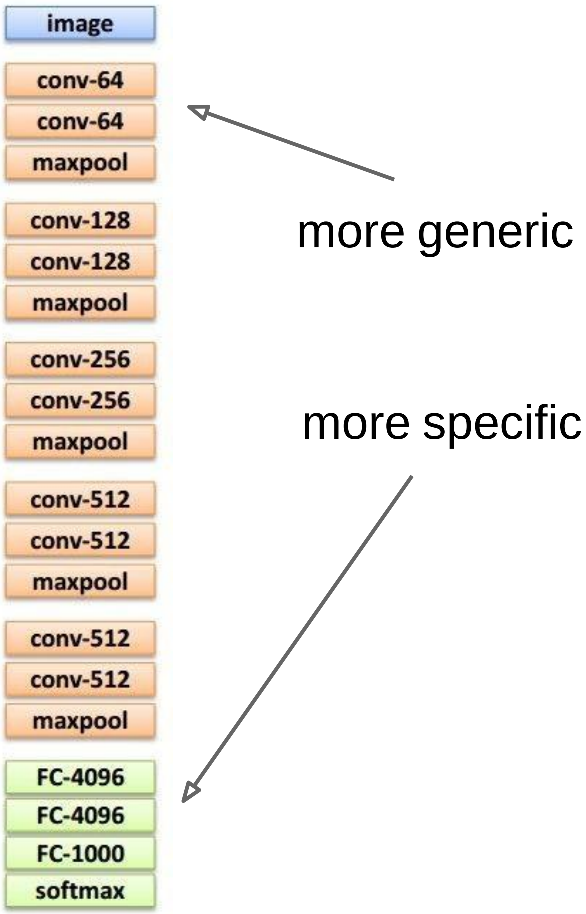
Source: e.g. classification of animals   Target: e.g. classification of cars



Another option: use network as feature extractor, train SVM/LR on extracted features for target task

Adapted from Andrej Karpathy

Transfer learning

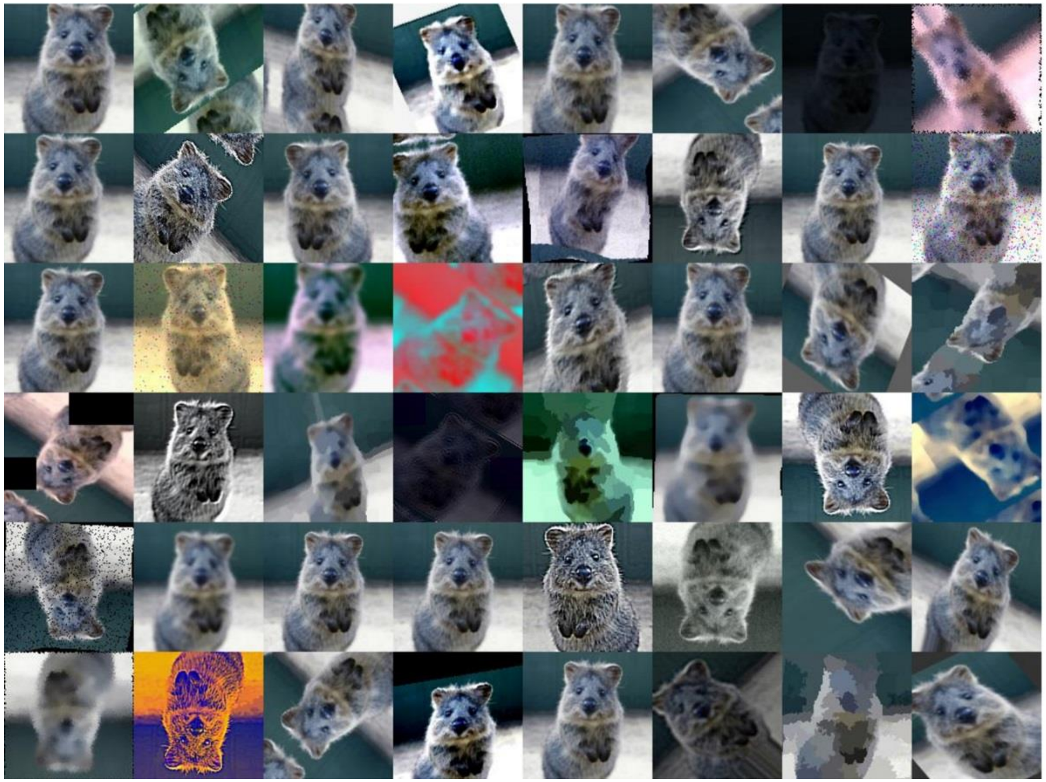


	very similar dataset	**very different dataset**
very little data	Use linear classifier on top layer	You’re in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Another solution: Data augmentation

Create *virtual* training samples; if images:

- Horizontal flip
- Random crop
- Color casting
- Geometric distortion



Jia-bin Huang, Image: https://github.com/aleju/imgaug

Packages

TensorFlow

Torch / PyTorch

Keras

Caffe and Caffe Model Zoo

Learning Resources

http://deeplearning.net/

http://cs231n.stanford.edu (CNNs, vision)

http://cs224d.stanford.edu/ (RNNs, language)

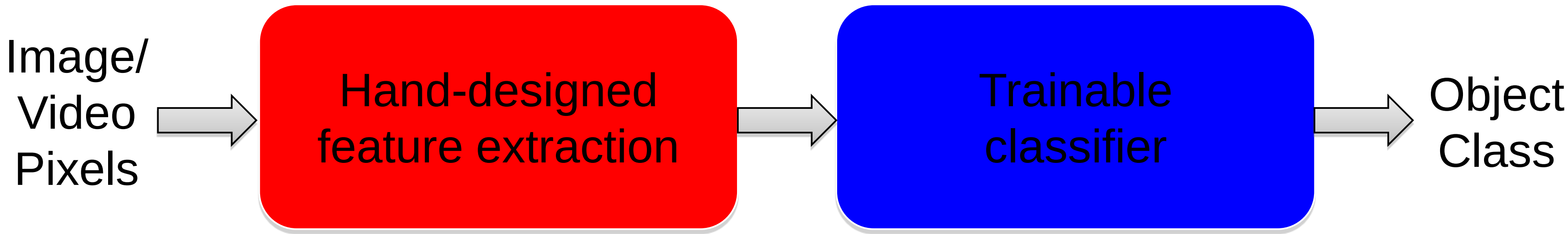
Summary

- Feed-forward network architecture
- Training deep neural nets
  - We need an objective function that measures and guides us towards good performance
  - We need a way to minimize the loss function: (stochastic, mini-batch) gradient descent
  - We need backpropagation to propagate error towards all layers and change weights at those layers
- Practices for preventing overfitting, training with little data

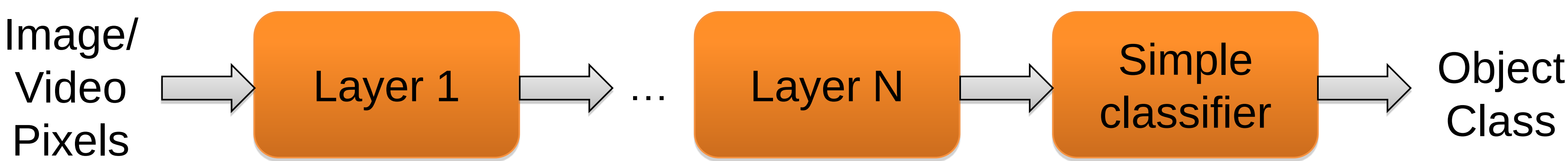
Convolutional Neural Networks

“Shallow” vs. “deep” vision architectures

Traditional recognition: “Shallow” architecture

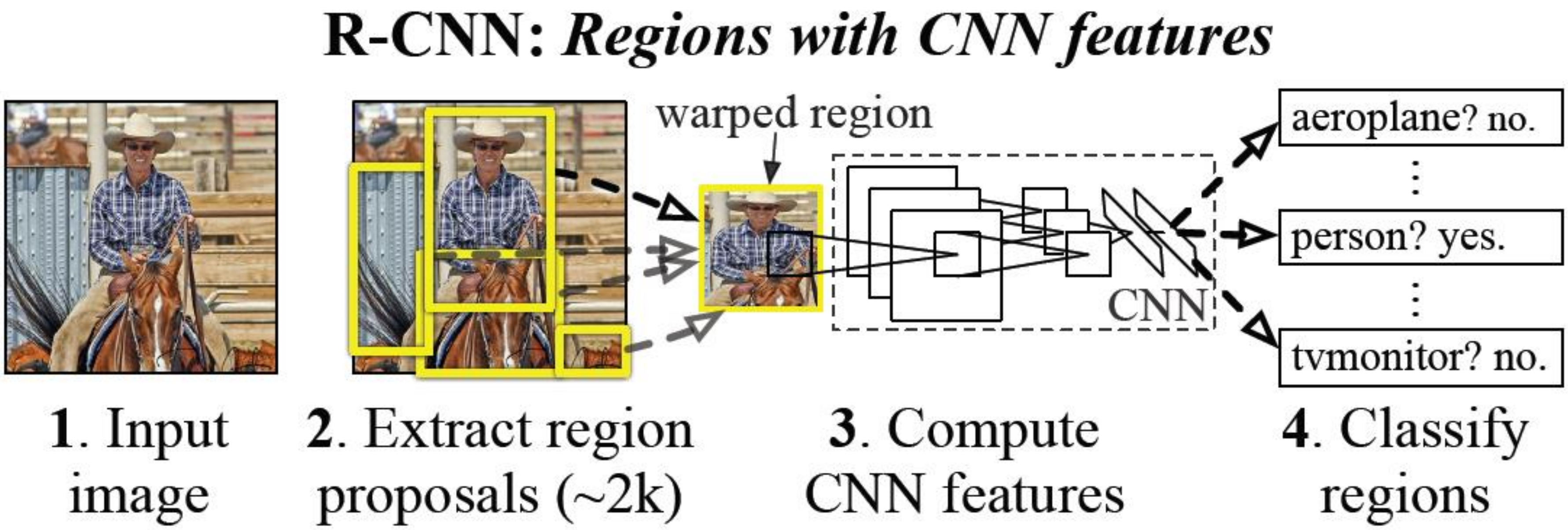


Deep learning: “Deep” architecture



Lana Lazebnik

Example: CNN features for detection

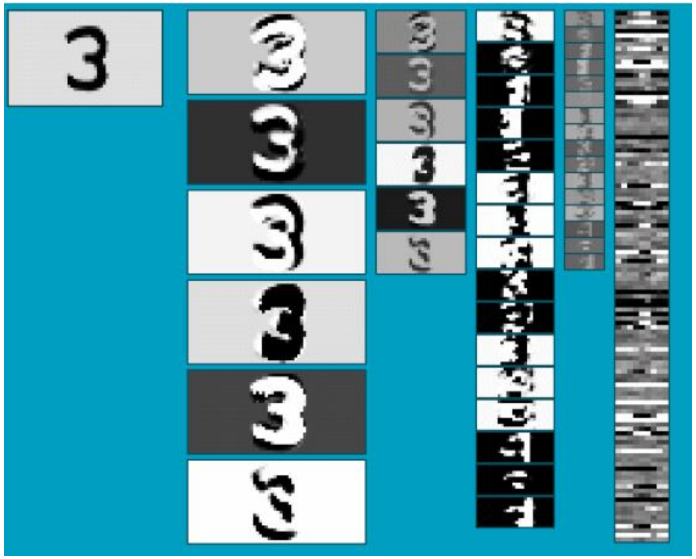


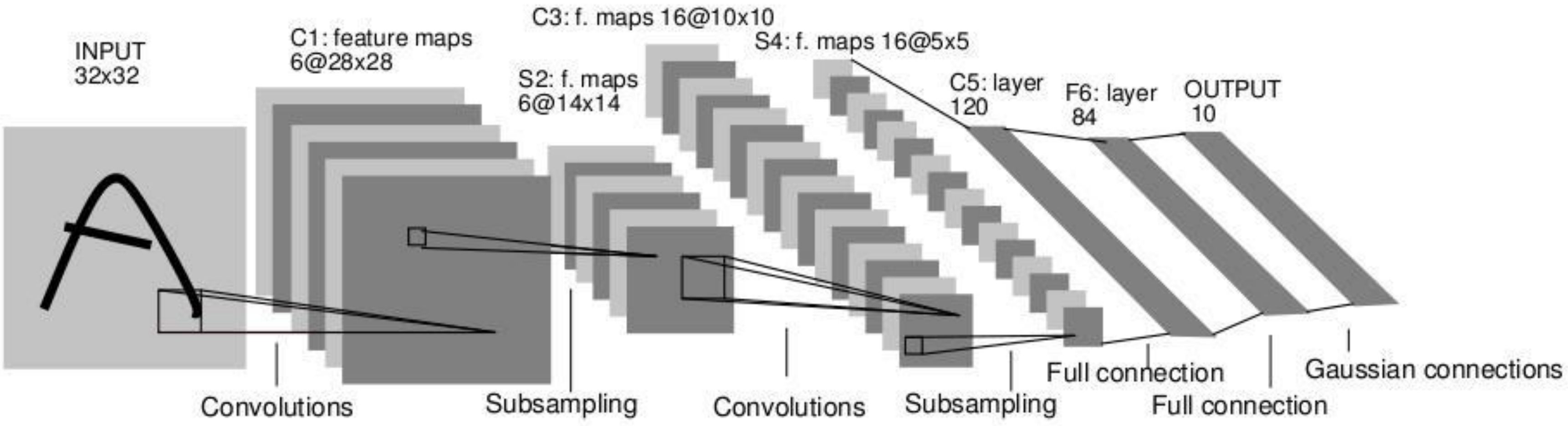
Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. **R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010**. For comparison, Uijlings et al. (2013) **report 35.1% mAP** using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. **The popular deformable part models perform at 33.4%.**

R. Girshick, J. Donahue, T. Darrell, and J. Malik, **Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation**, CVPR 2014.

Convolutional Neural Networks (CNN)

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant, *more abstract* features
- Classification layer at the end





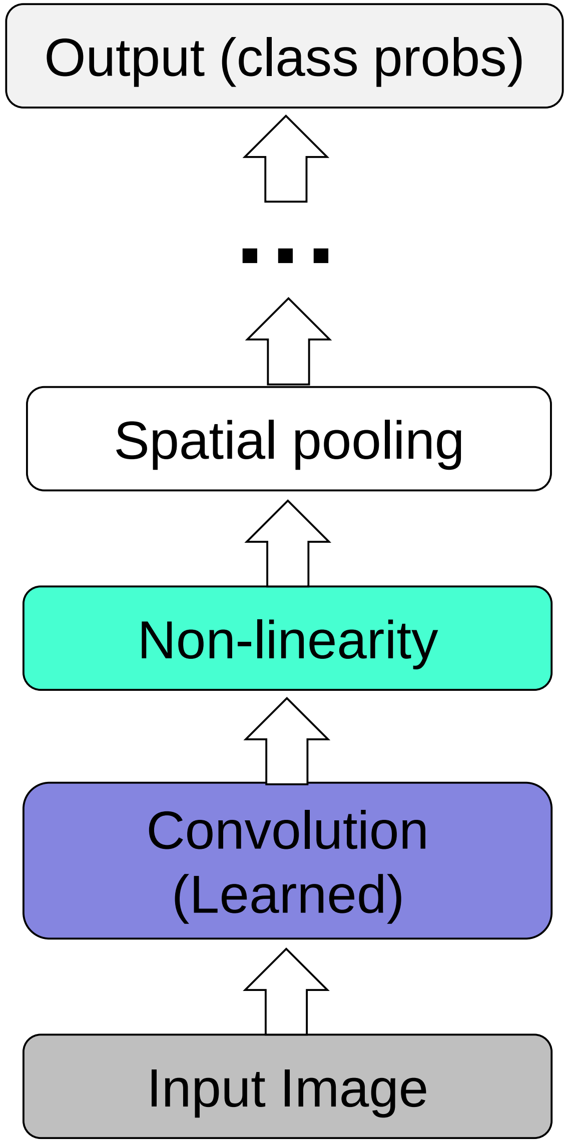
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Adapted from Rob Fergus

Convolutional Neural Networks (CNN)

- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Apply non-linearity
  3. Spatial pooling (downsample)

- Supervised training of convolutional filters by back-propagating classification error

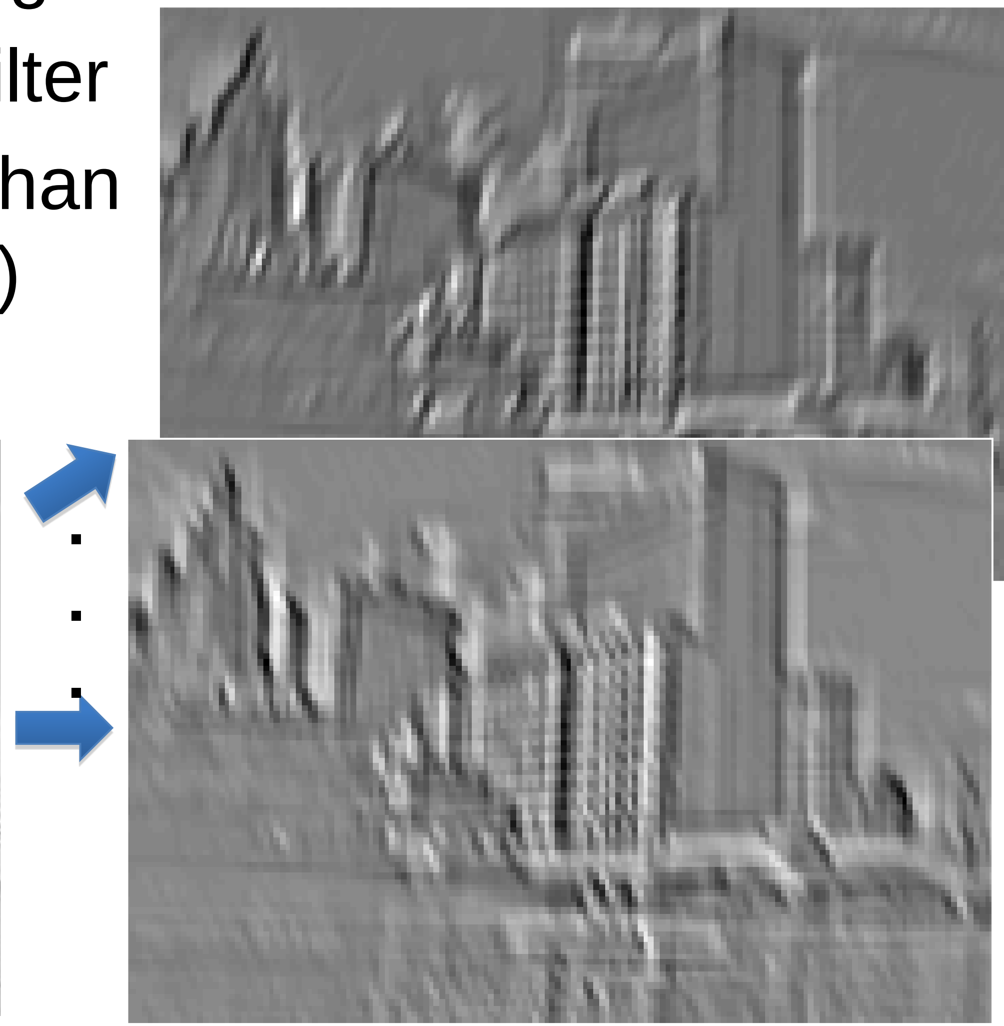


Adapted from Lana Lazebnik

1. Convolution

- Apply learned filter weights
- One feature map per filter
- Stride can be greater than 1 (faster, less memory)

Input

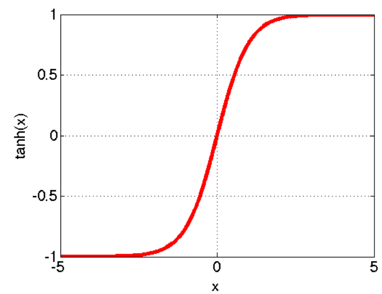


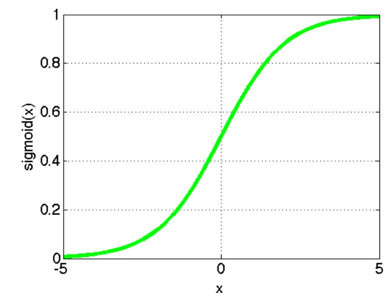
Feature Map

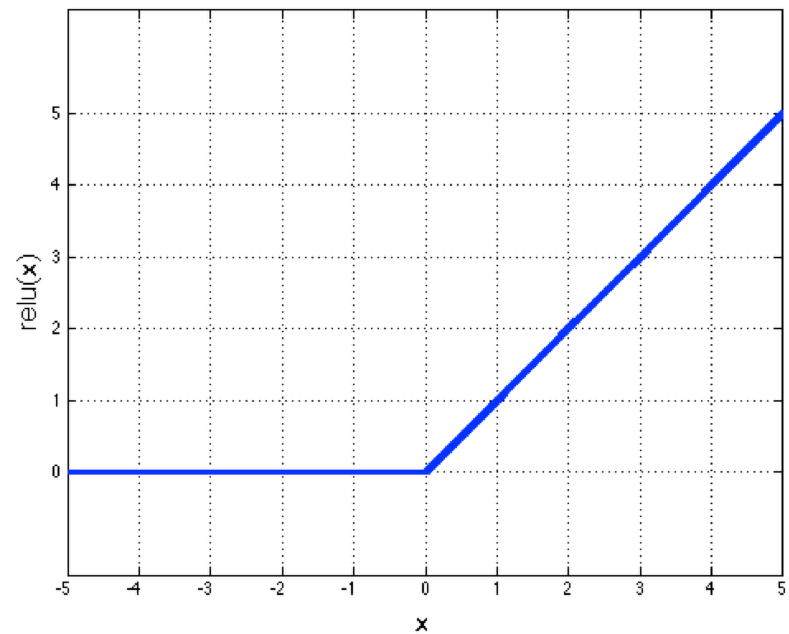
Adapted from Rob Fergus

2. Non-Linearity

- Per-element (independent)
- Options:
  - Tanh
  - Sigmoid
  - Rectified linear unit (ReLU)
    - – Avoids saturation issues



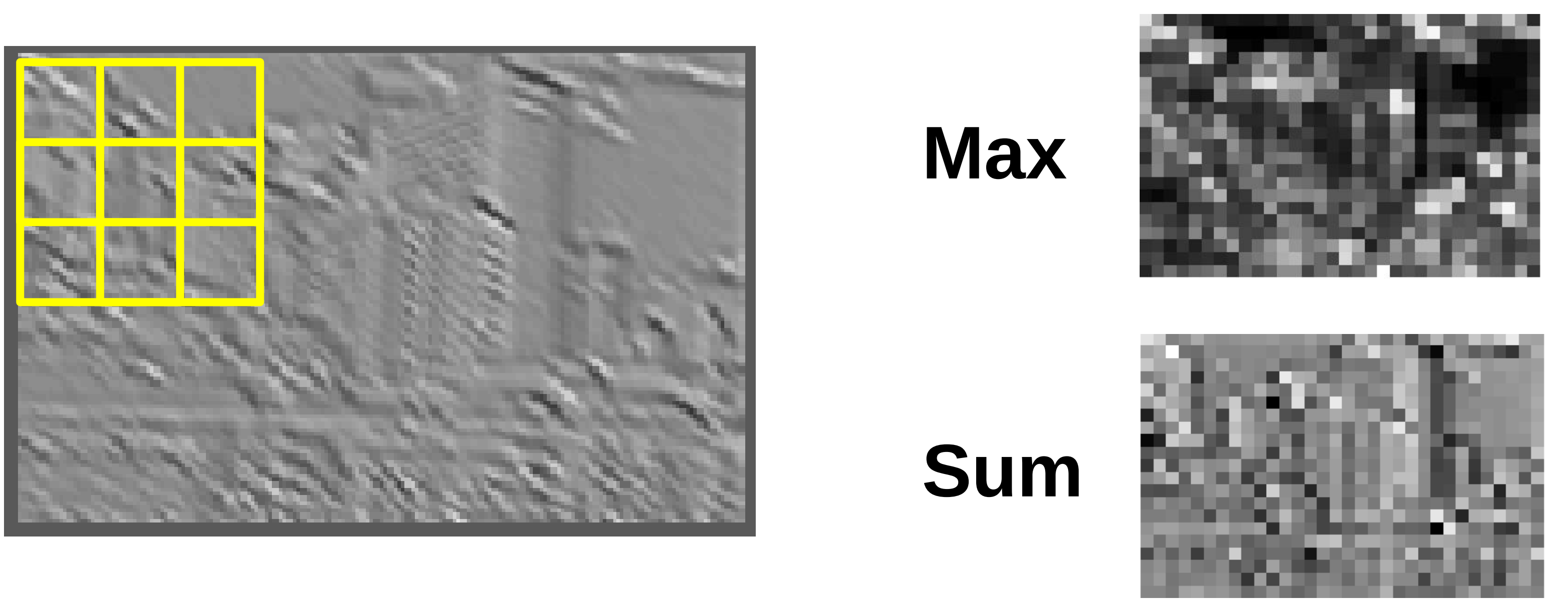




Adapted from Rob Fergus

3. Spatial Pooling

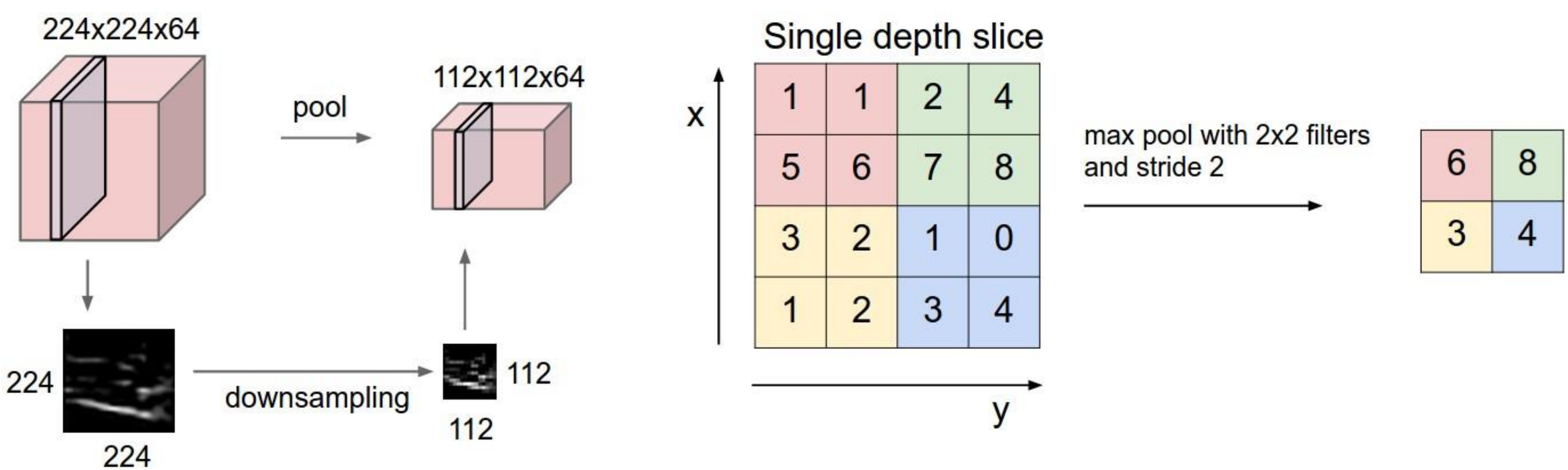
- Sum or max over non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)



Adapted from Rob Fergus

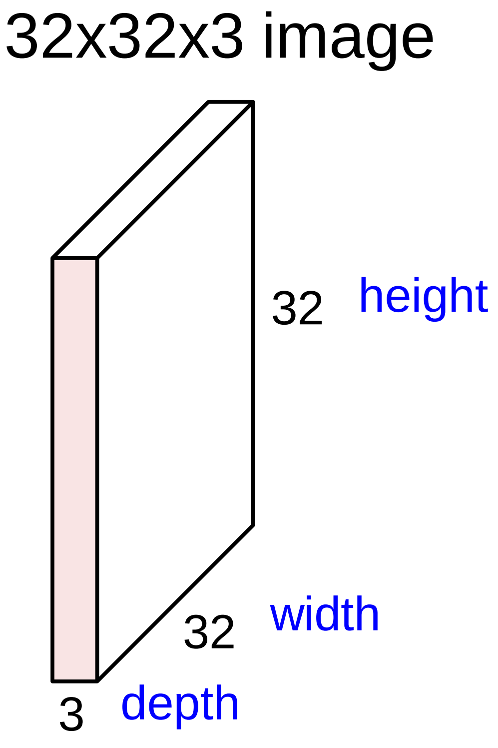
3. Spatial Pooling

- Sum or max over non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)



Rob Fergus, figure from Andrej Karpathy

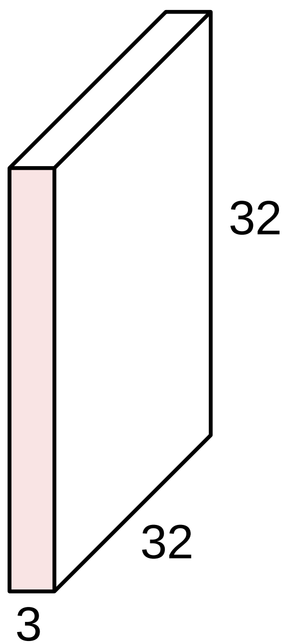
Convolutions: More detail



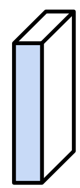
Andrej Karpathy

Convolutions: More detail

32x32x3 image



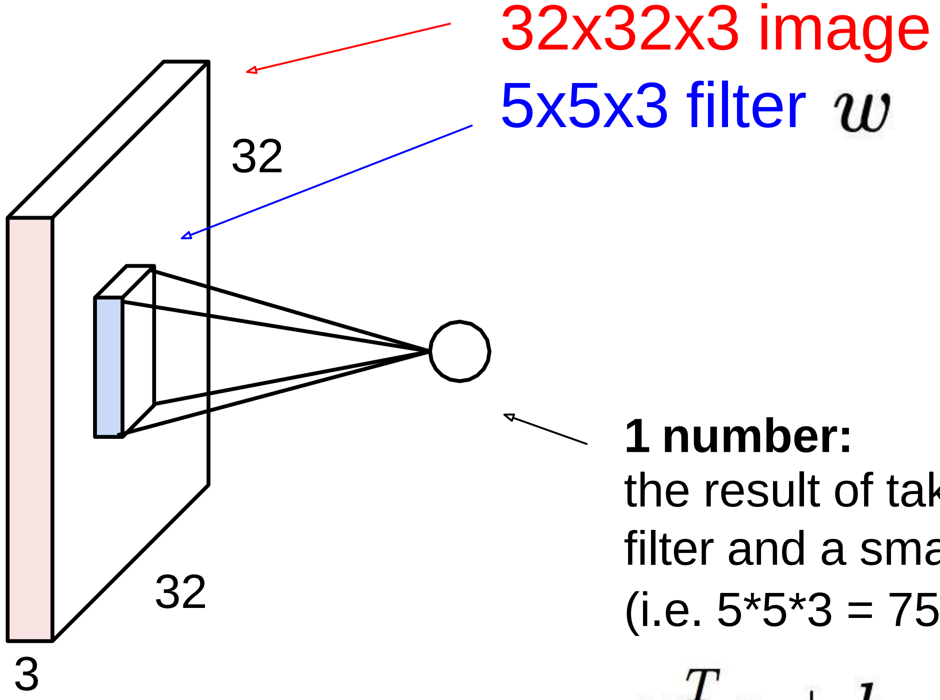
5x5x3 filter



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

Convolutions: More detail

Convolution Layer

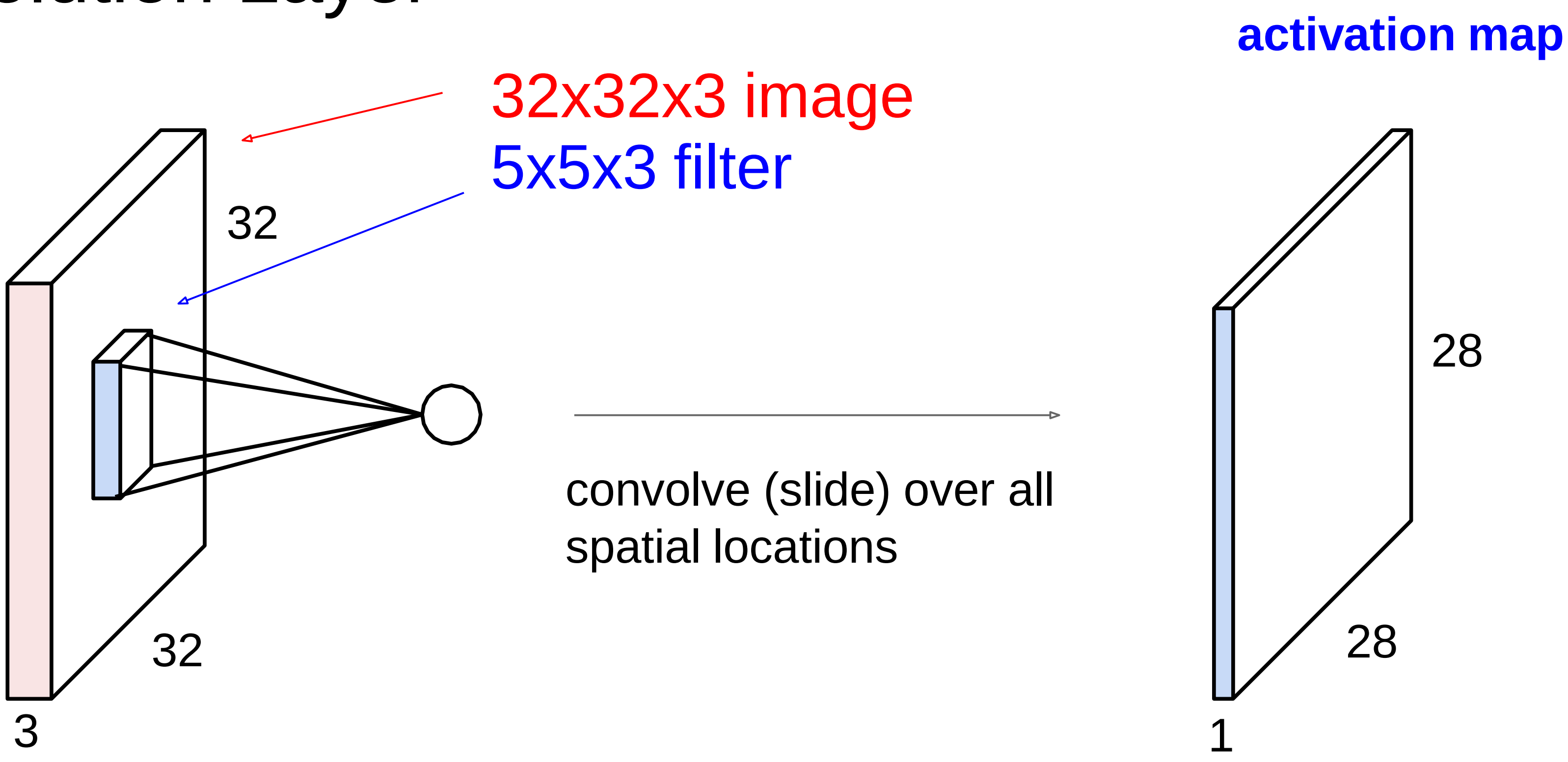


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolutions: More detail

Convolution Layer



Andrej Karpathy

Convolutions: More detail

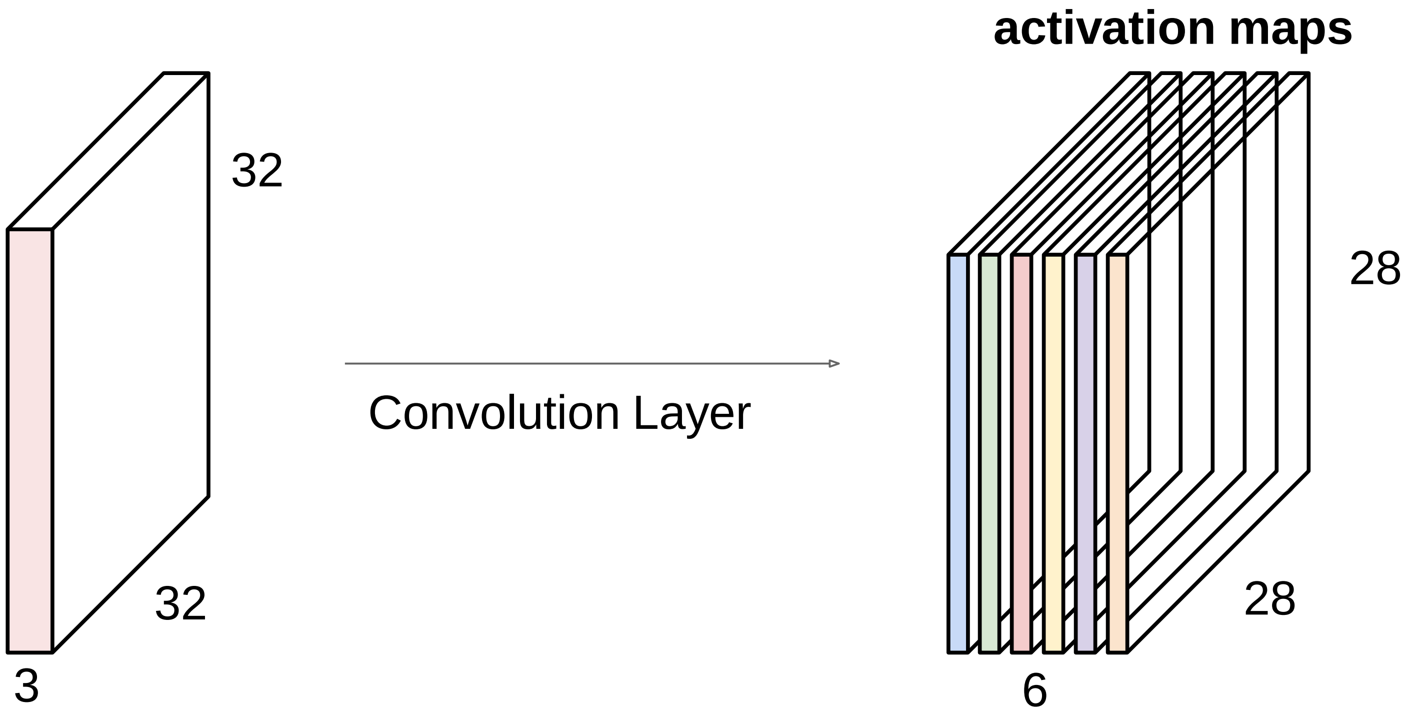
Convolution Layer

consider a second, green filter

Andrej Karpathy

Convolutions: More detail

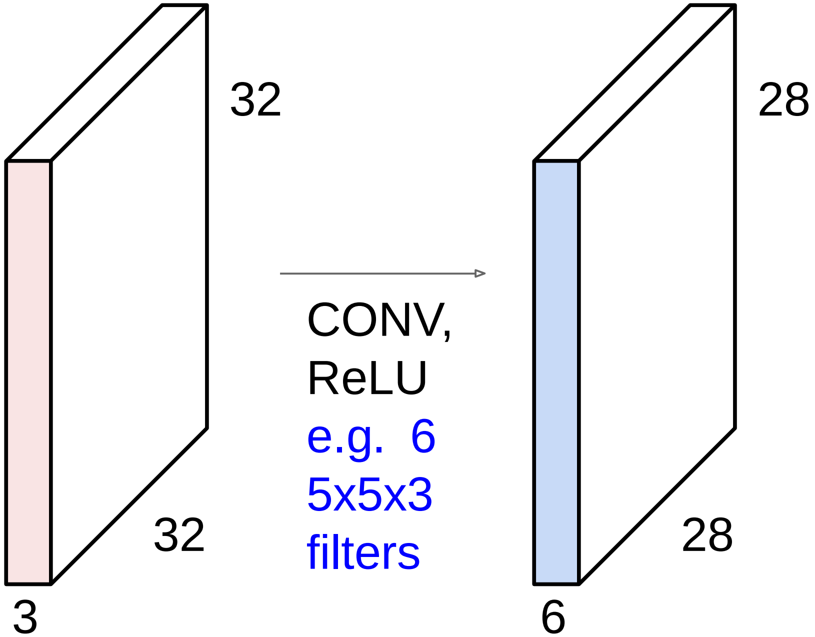
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

Convolutions: More detail

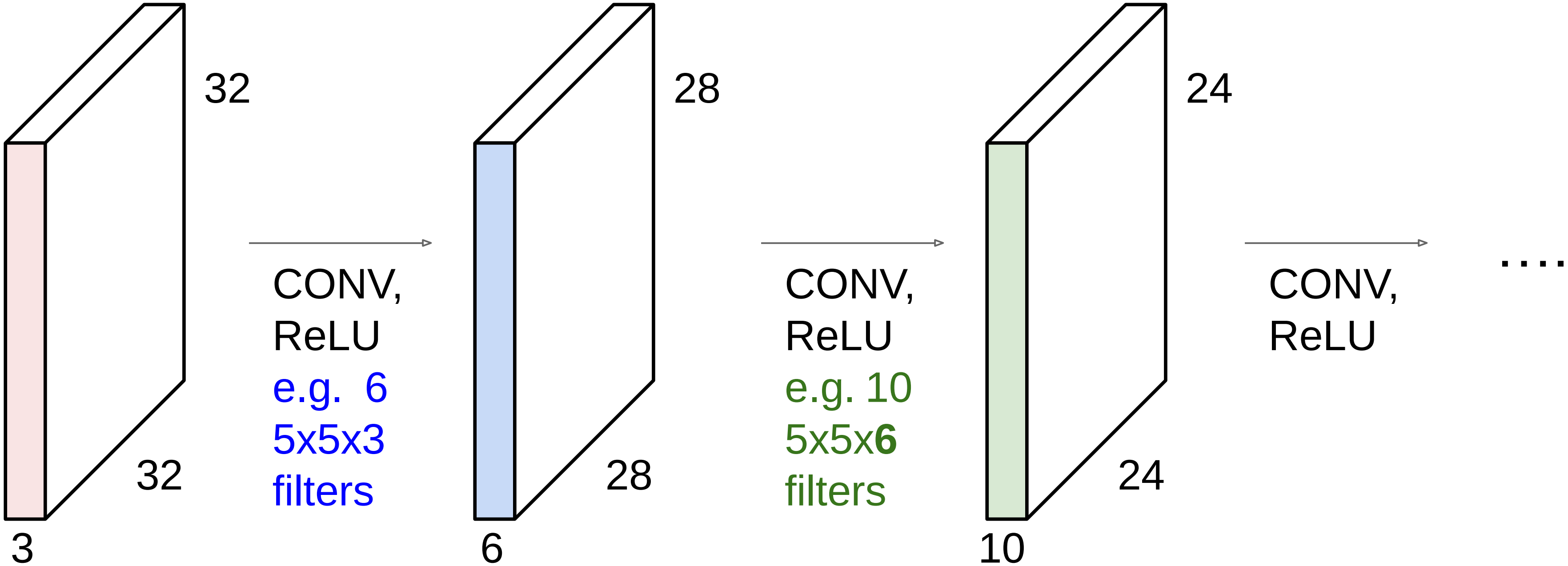
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Andrej Karpathy

Convolutions: More detail

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

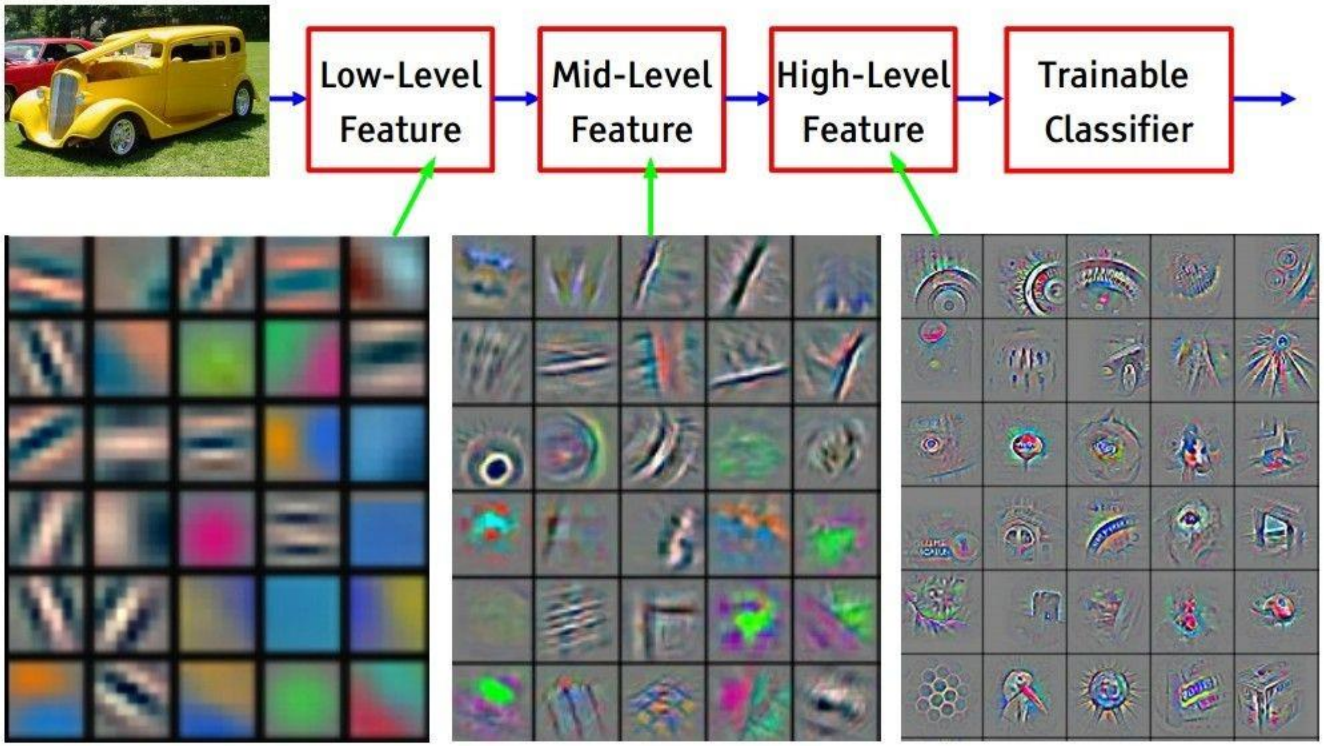


Andrej Karpathy

Convolutions: More detail

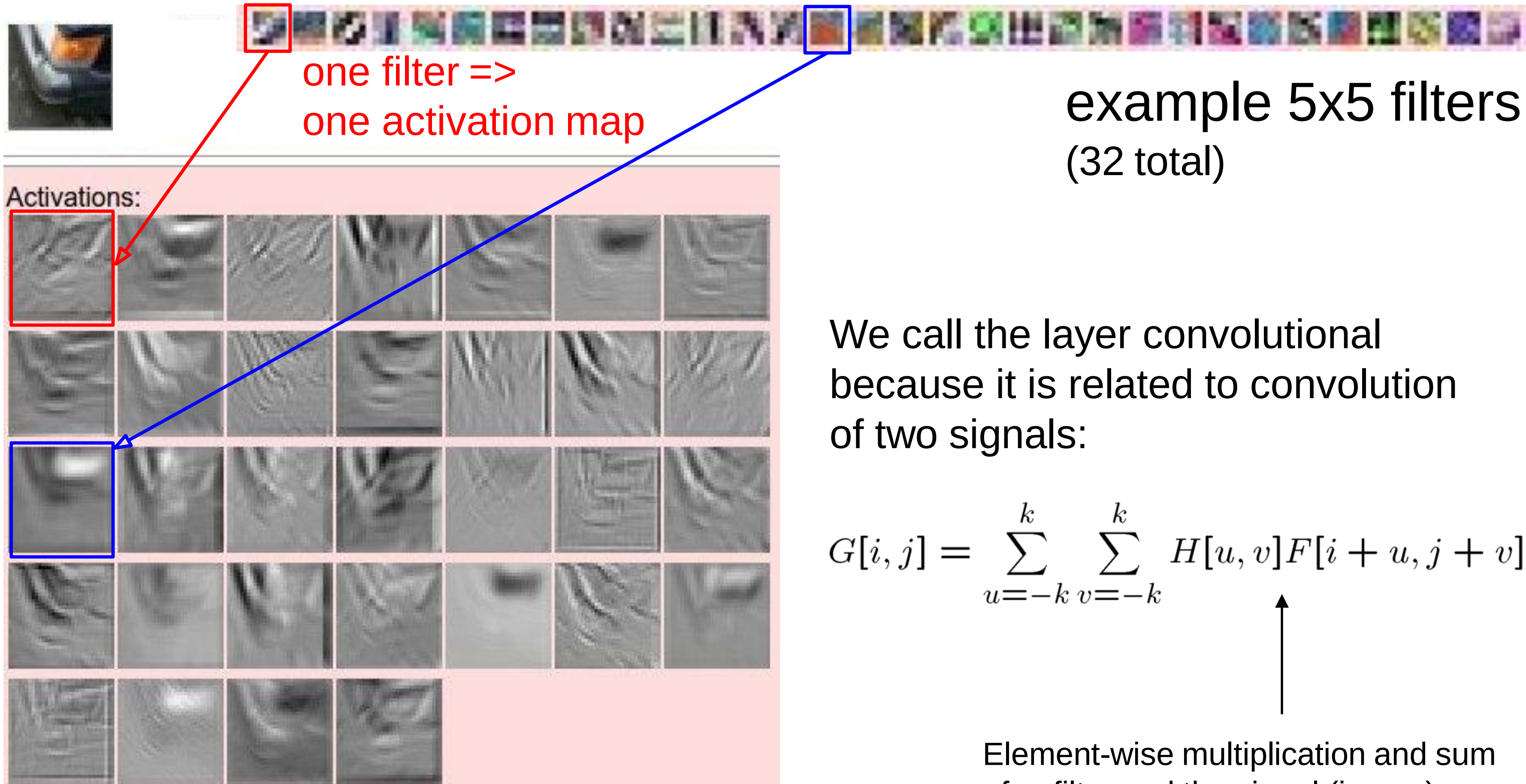
Preview

[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutions: More detail



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

Element-wise multiplication and sum of a filter and the signal (image)

Adapted from Andrej Karpathy, Kristen Grauman

The First Popular Architecture: AlexNet

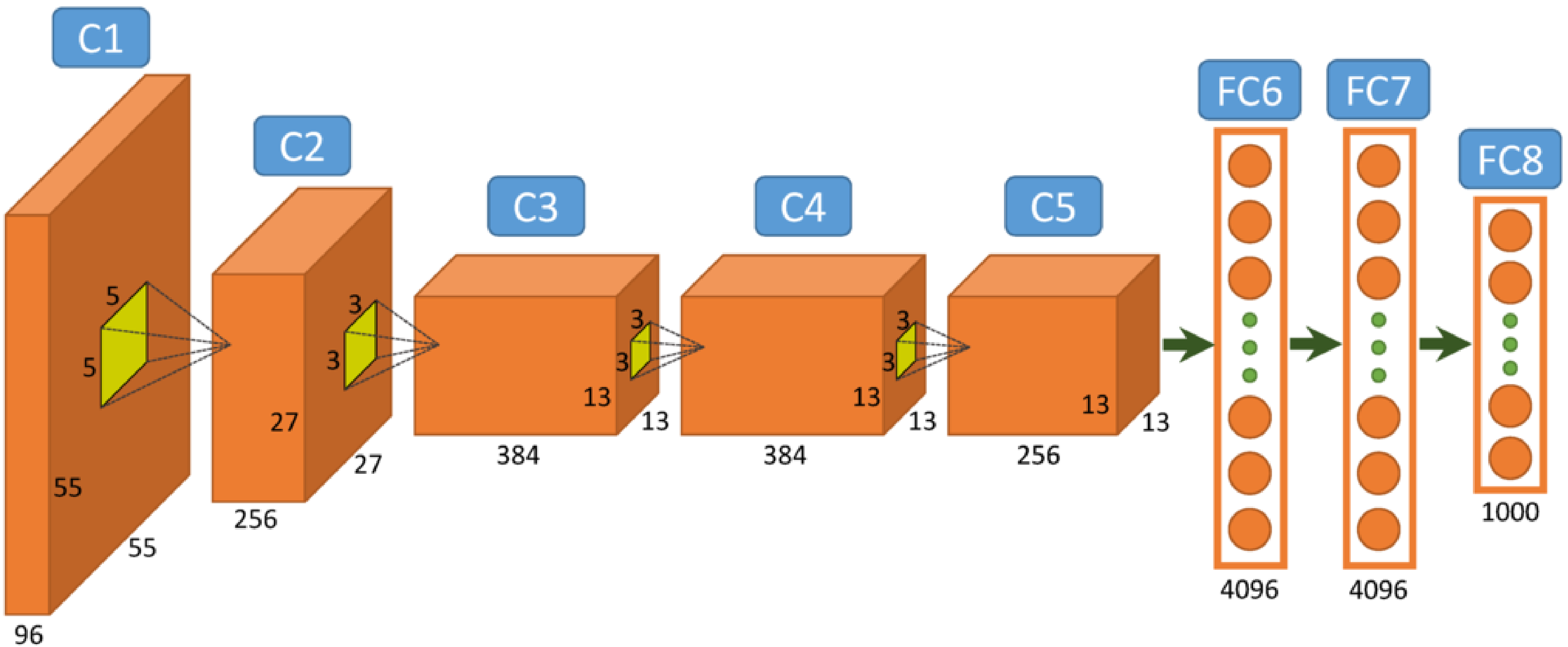
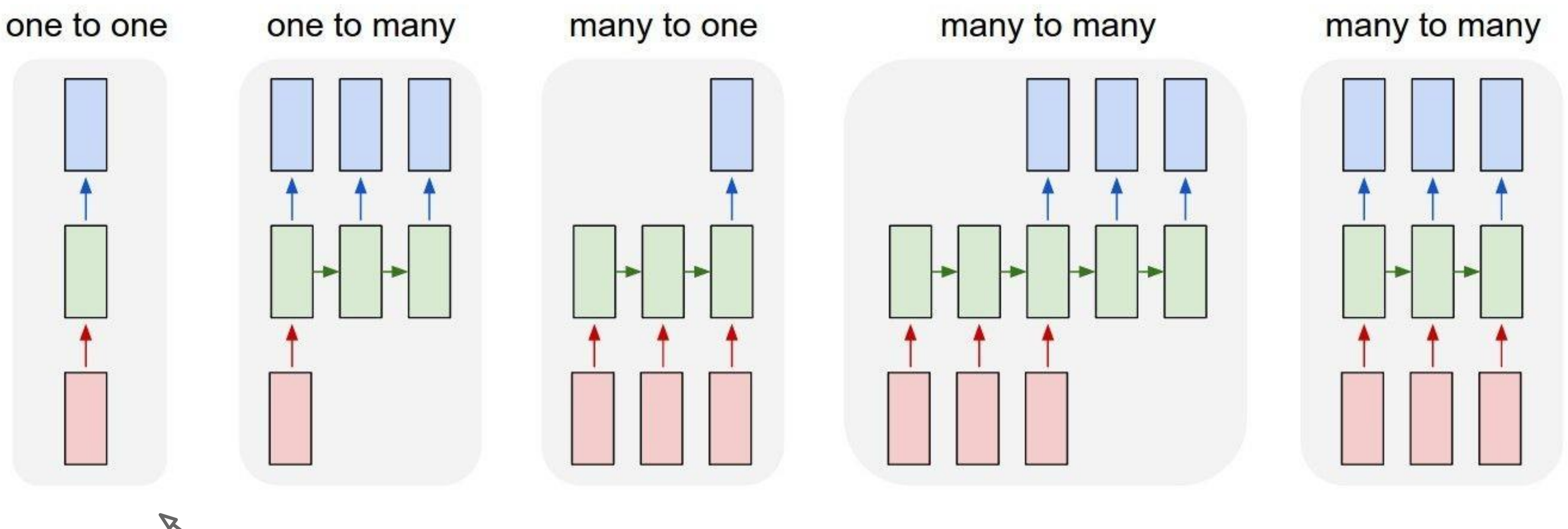


Figure from http://www.mdpi.com/2072-4292/7/11/14680/htm

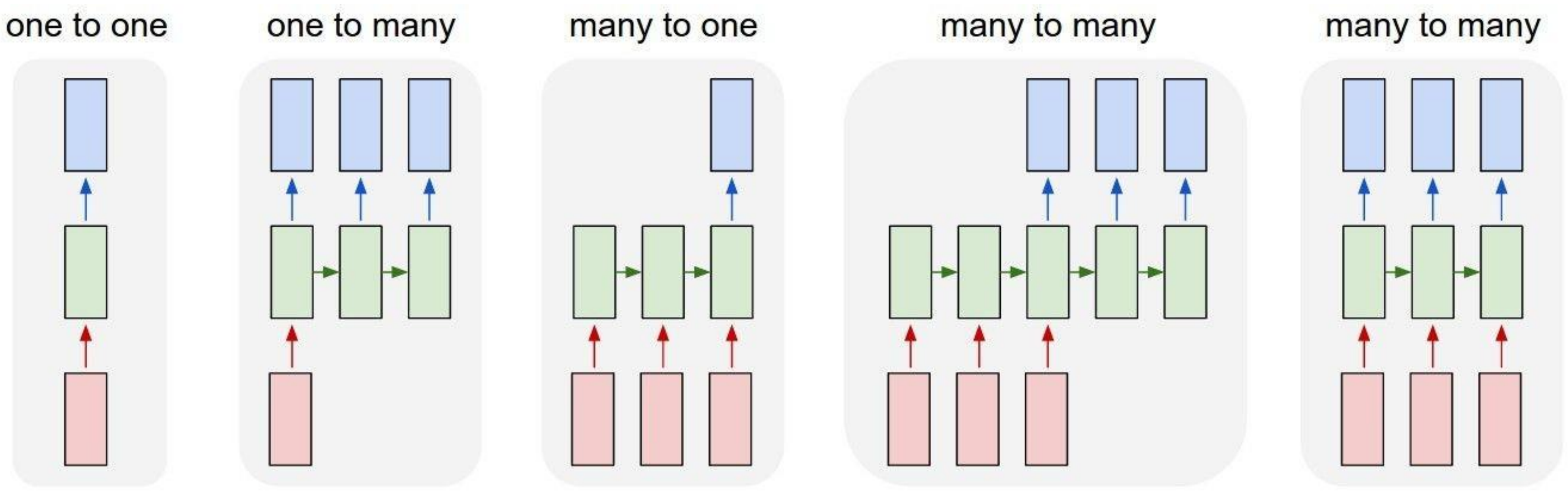
Recurrent Neural Networks

Examples of Recurrent Networks



vanilla neural networks

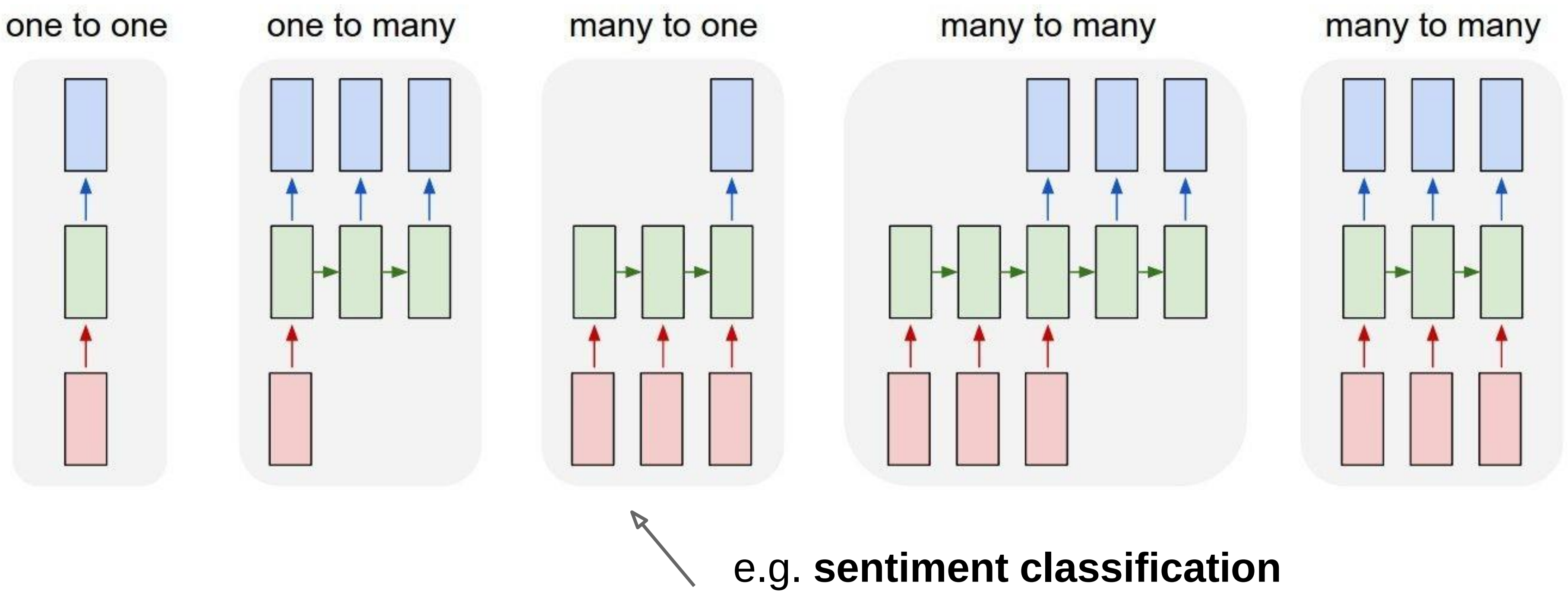
Examples of Recurrent Networks



e.g. **image captioning**
image -> sequence of words

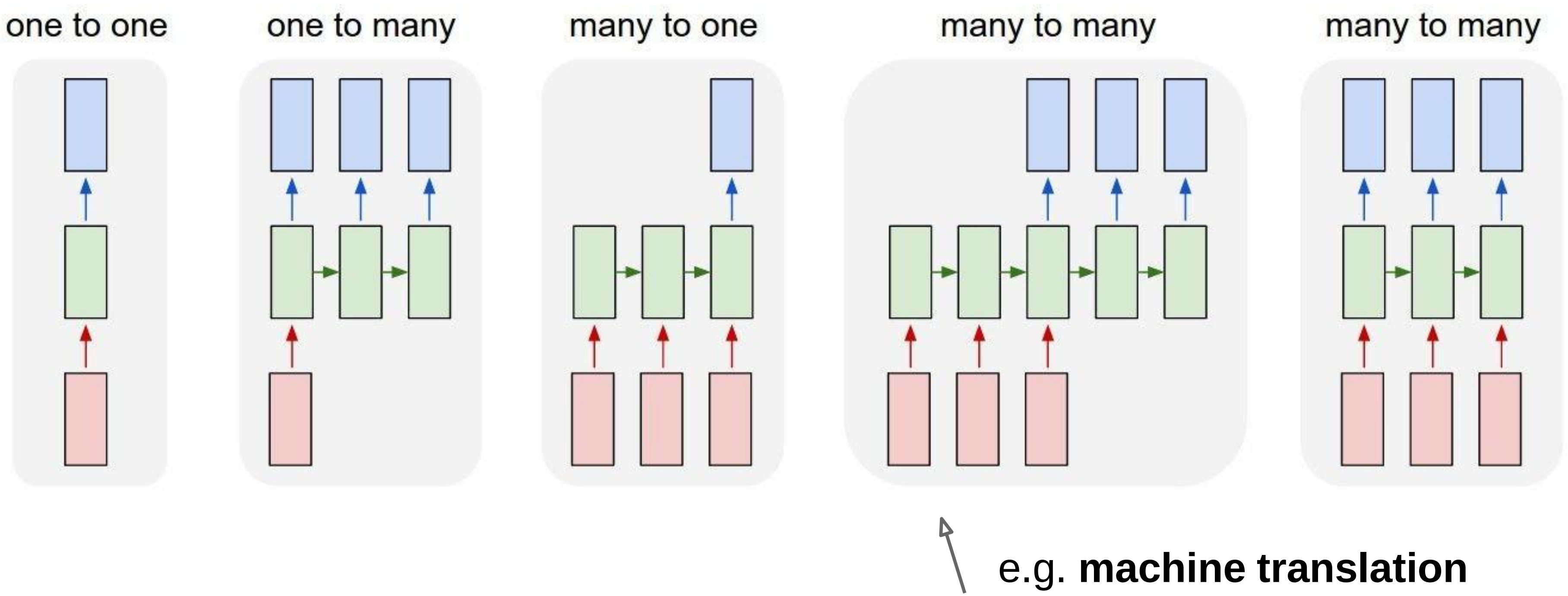
Andrej Karpathy

Examples of Recurrent Networks



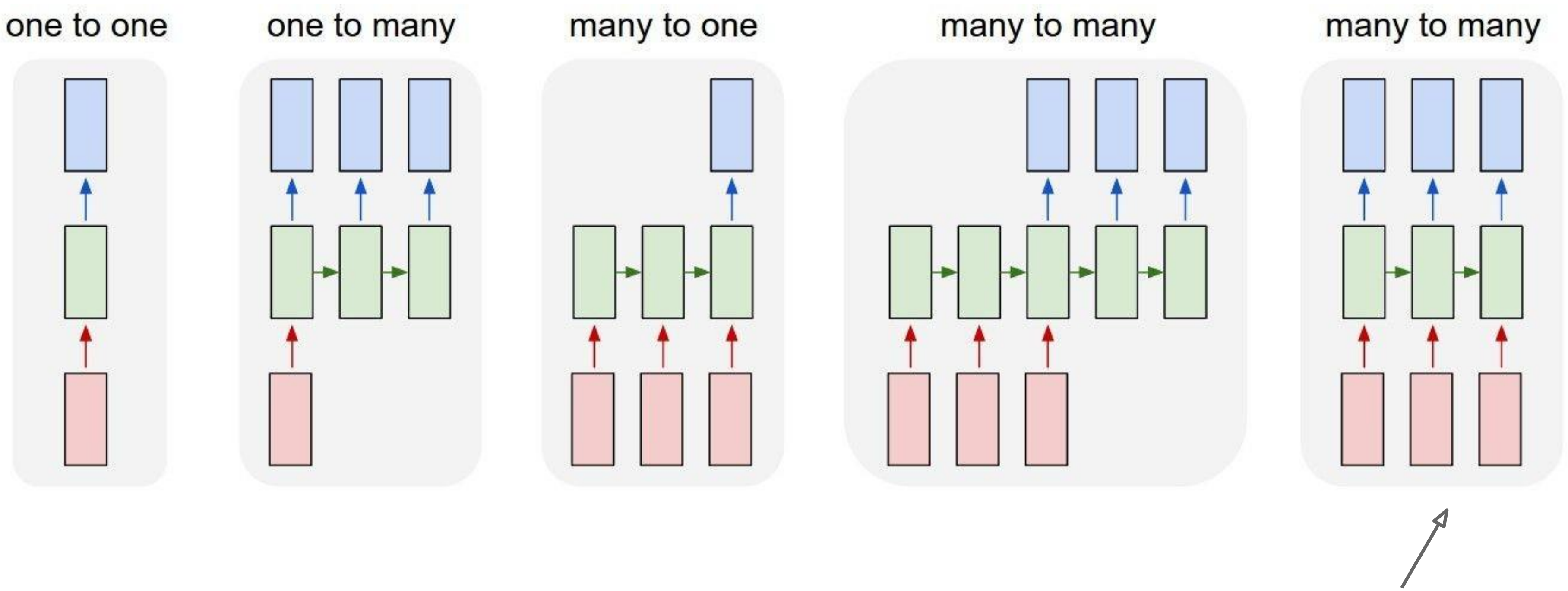
e.g. **sentiment classification**
sequence of words -> sentiment

Examples of Recurrent Networks



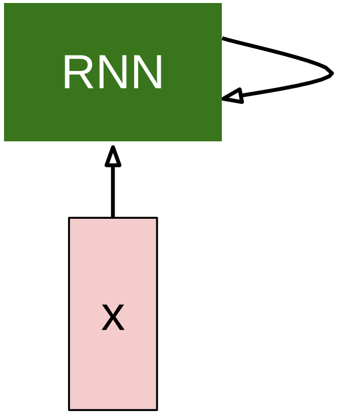
e.g. **machine translation**
seq of words -> seq of words

Examples of Recurrent Networks

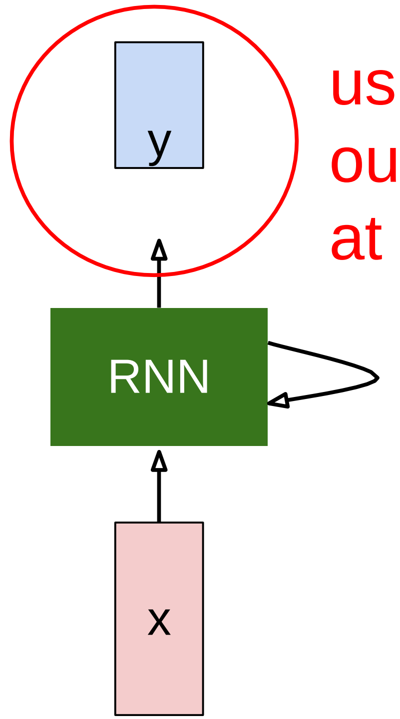


e.g. **video classification on frame level**

Recurrent Neural Network



Recurrent Neural Network



usually want to output a prediction at some time steps

Adapted from Andrej Karpathy

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

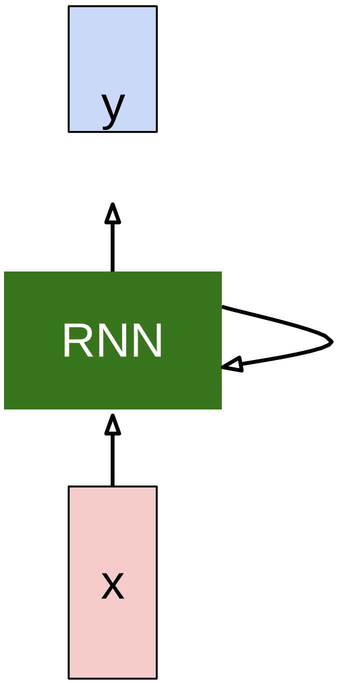
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

input vector at some time step



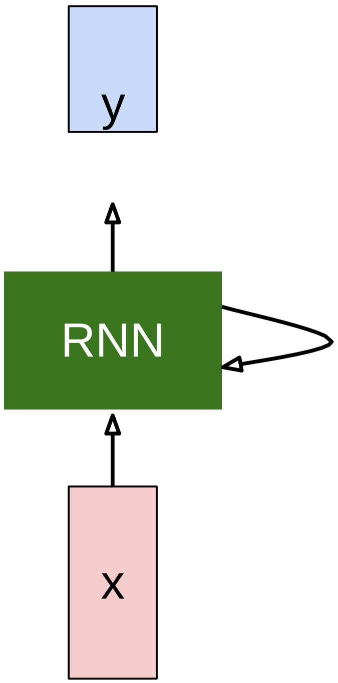
Andrej Karpathy

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

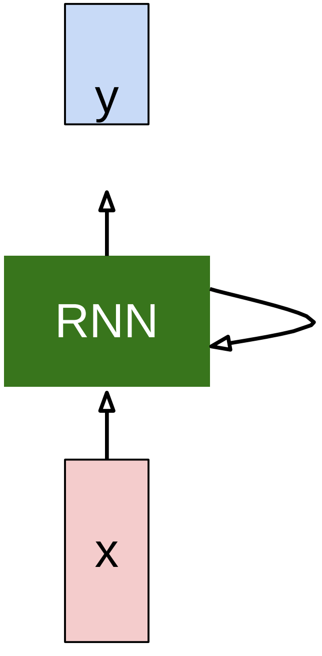
$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.



(Vanilla) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

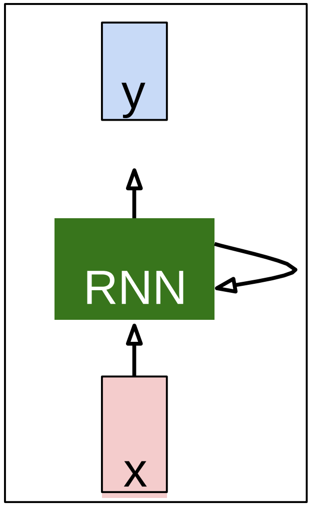
$$y_t = W_{hy}h_t$$

Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

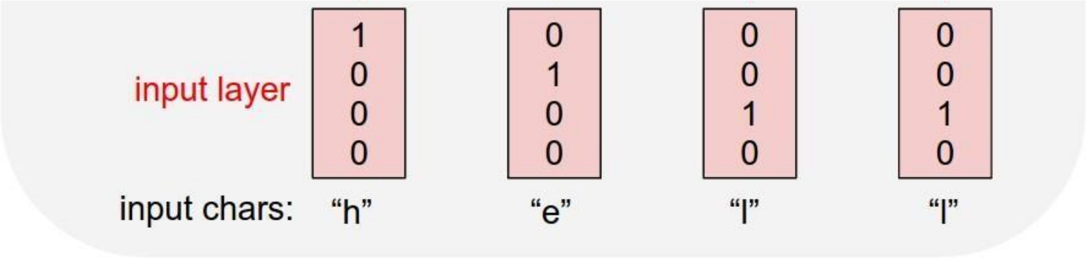


Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



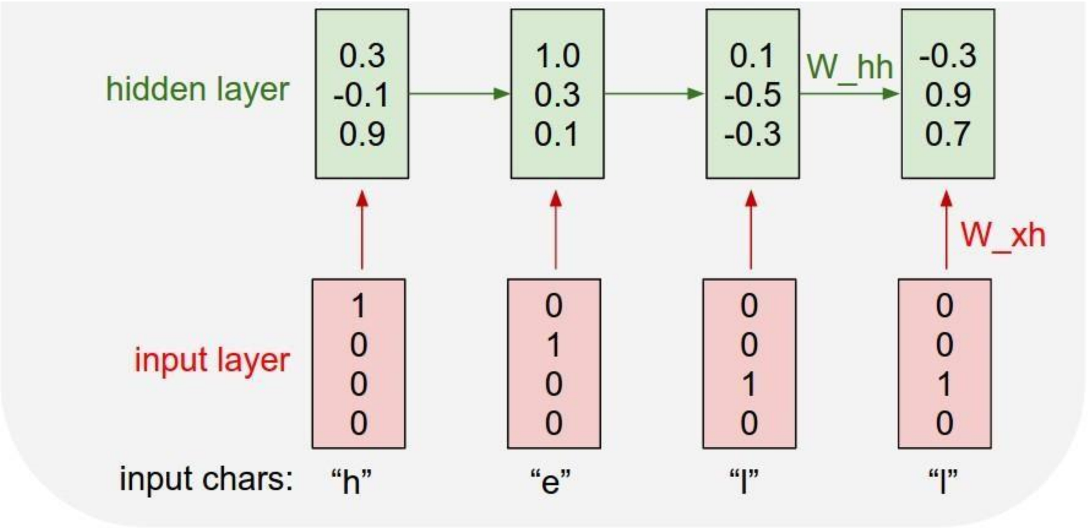
Example

Character-level language model example

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

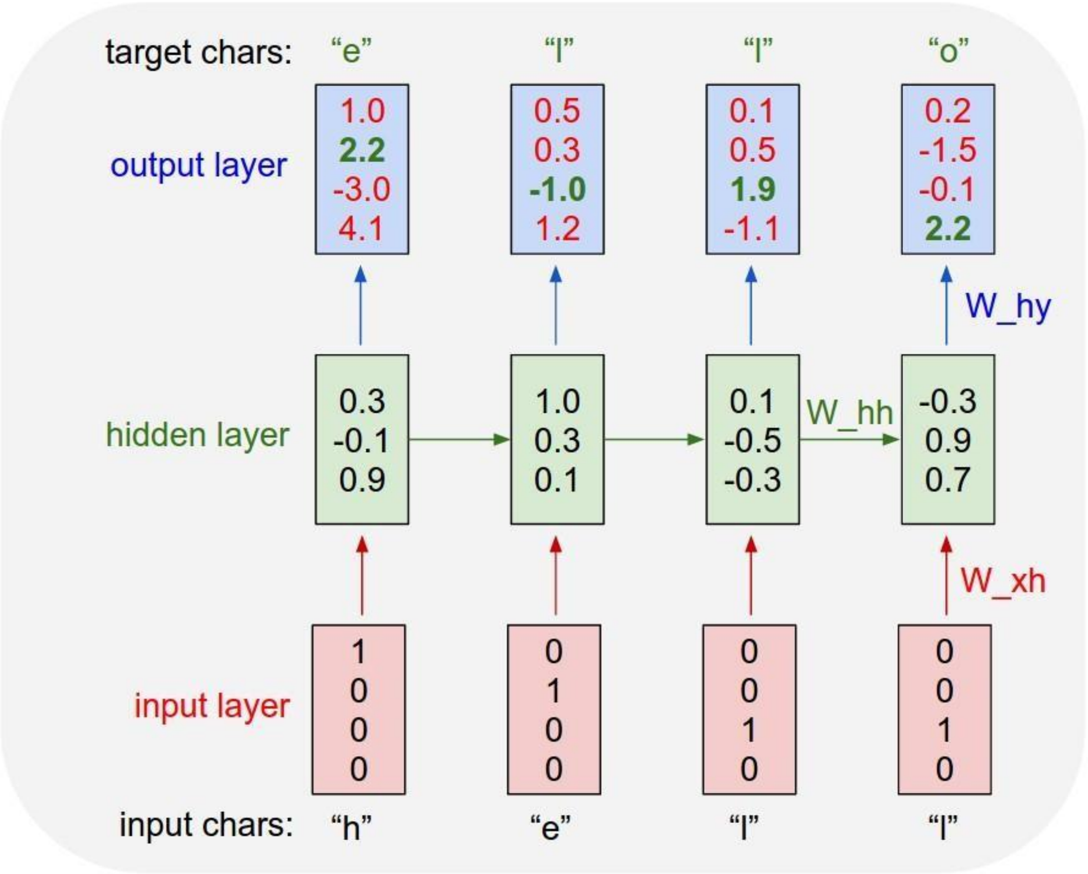


Example

Character-level language model example

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



Extensions

- Vanishing gradient problem makes it hard to model long sequences
  - Multiplying together many values between 0 and 1 (range of gradient of sigmoid, tanh)
- One solution: Use RELU
- Another solution: Use RNNs with gates
  - Adaptively decide how much of memory to keep
  - Gated Recurrent Units (GRUs), Long Short Term Memories (LSTMs)

Generating poetry with RNNs

Sonnet 116 – Let me not ...

by William Shakespeare

Let me not to the marriage of true minds
Admit impediments. Love is not love
Which alters when it alteration finds,
Or bends with the remover to remove:
O no! it is an ever-fixed mark
That looks on tempests and is never shaken;
It is the star to every wandering bark,
Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
Within his bending sickle's compass come:
Love alters not with his brief hours and weeks,
But bears it out even to the edge of doom.
If this be error and upon me proved,
I never writ, nor no man ever loved.

Andrej Karpathy

Generating poetry with RNNs

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

↓ train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

↓ train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

↓ train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

More info: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Andrej Karpathy

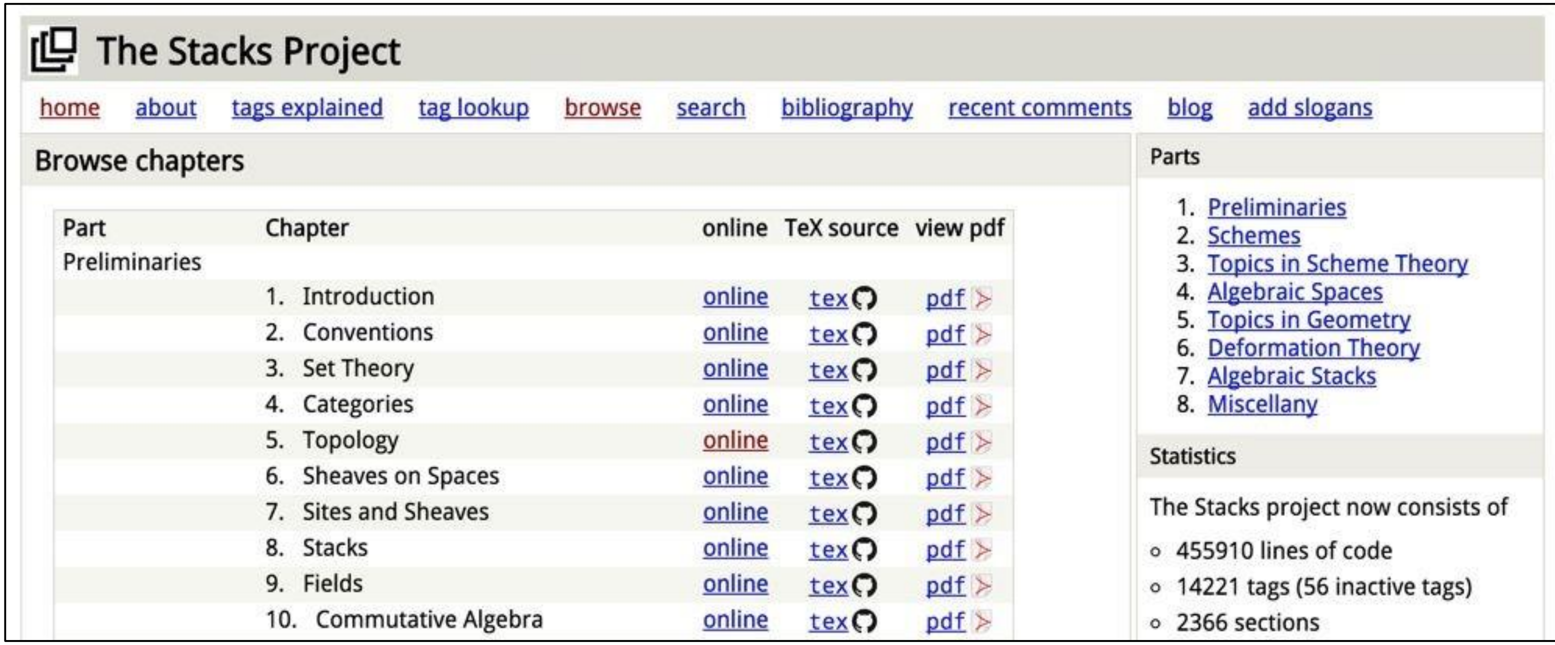
Generating poetry with RNNs

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

Generating textbooks with RNNs

open source textbook on algebraic geometry



Latex source

Andrej Karpathy

Generating textbooks with RNNs

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition* **??** *(without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation* **??**. *Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Andrej Karpathy

Generating textbooks with RNNs

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma **??**. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

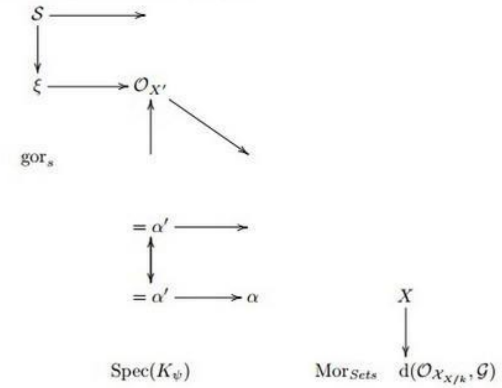
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \ \ \text{-1}(\mathcal{O}_{X_{étale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Andrej Karpathy

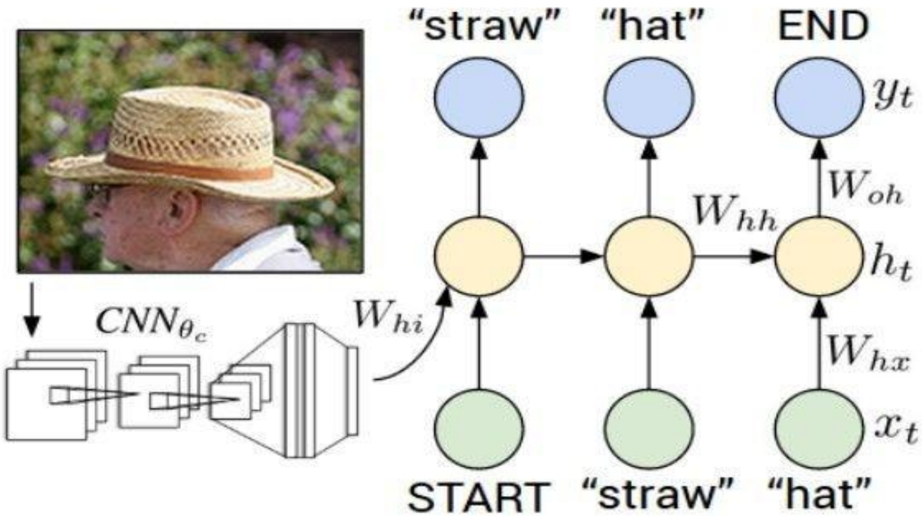
Generating code with RNNs

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated C code

Andrej Karpathy

Image Captioning

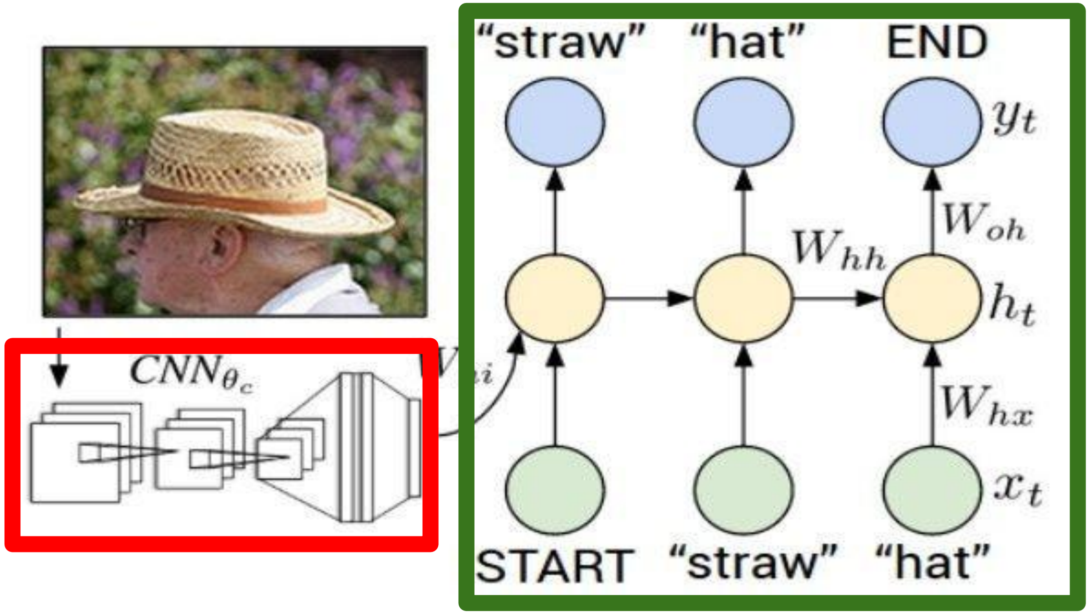


CVPR 2015:
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Adapted from Andrej Karpathy

Image Captioning

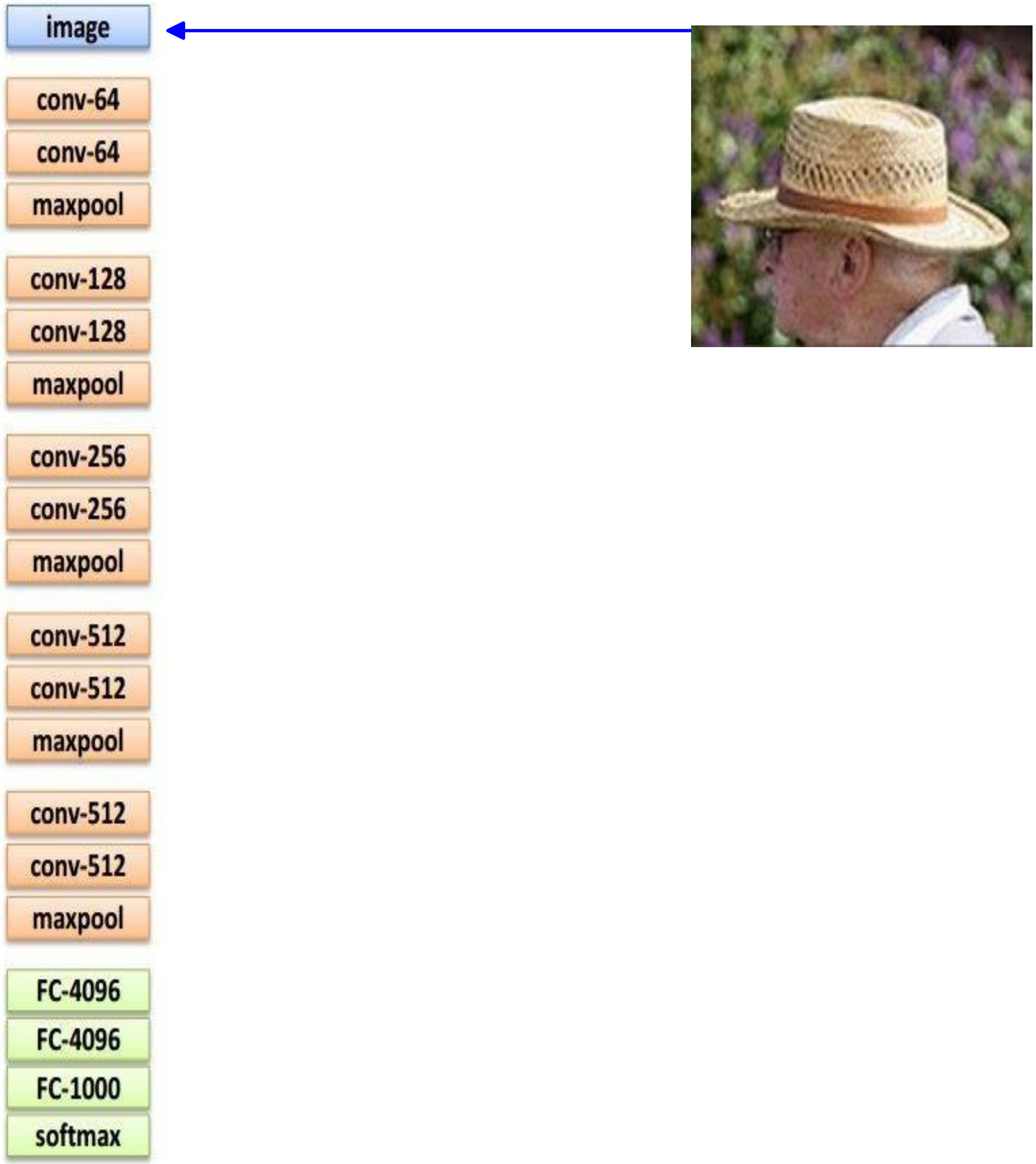
Recurrent Neural Network



Convolutional Neural Network

Image Captioning

test image



test image

Andrej Karpathy

test image

Andrej Karpathy

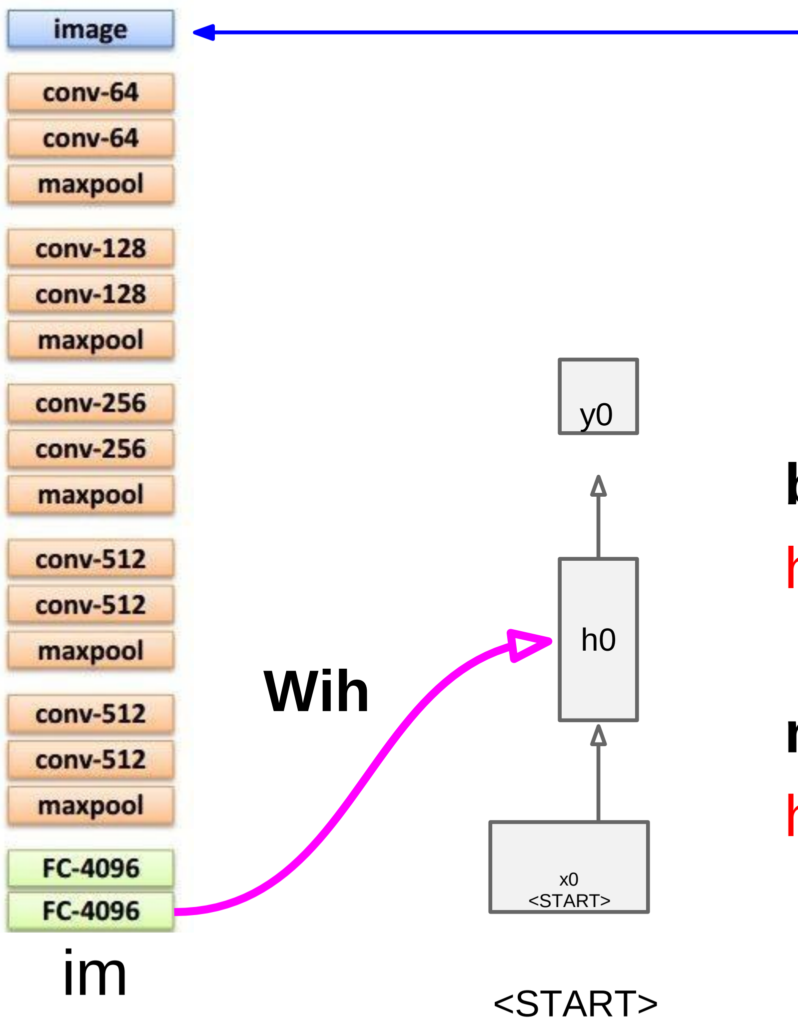
Image Captioning

test image

x0
<START>

<START>

Image Captioning



test image

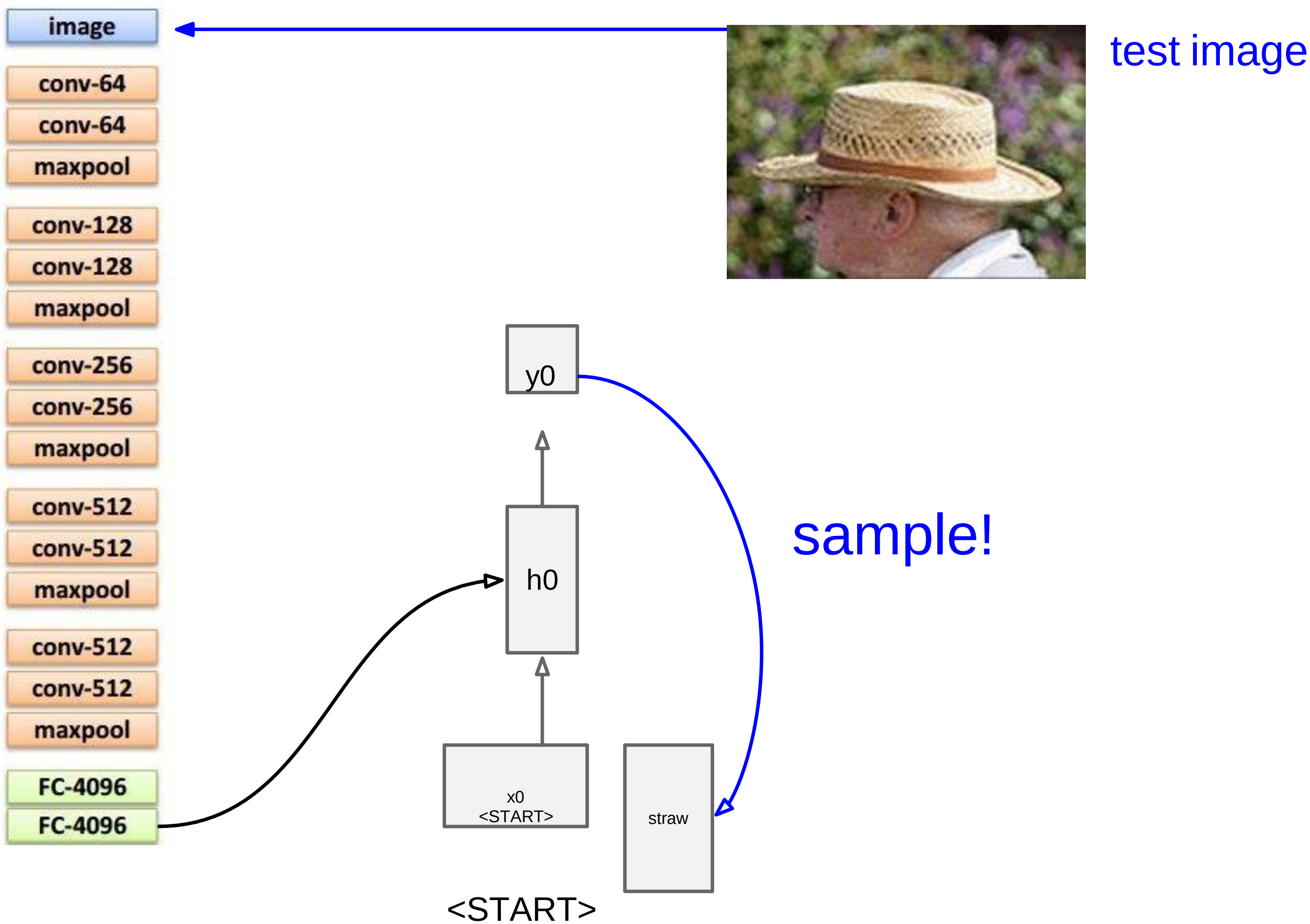
before:

$$h = \tanh(W_{xh} * x + W_{hh} * h)$$

now:

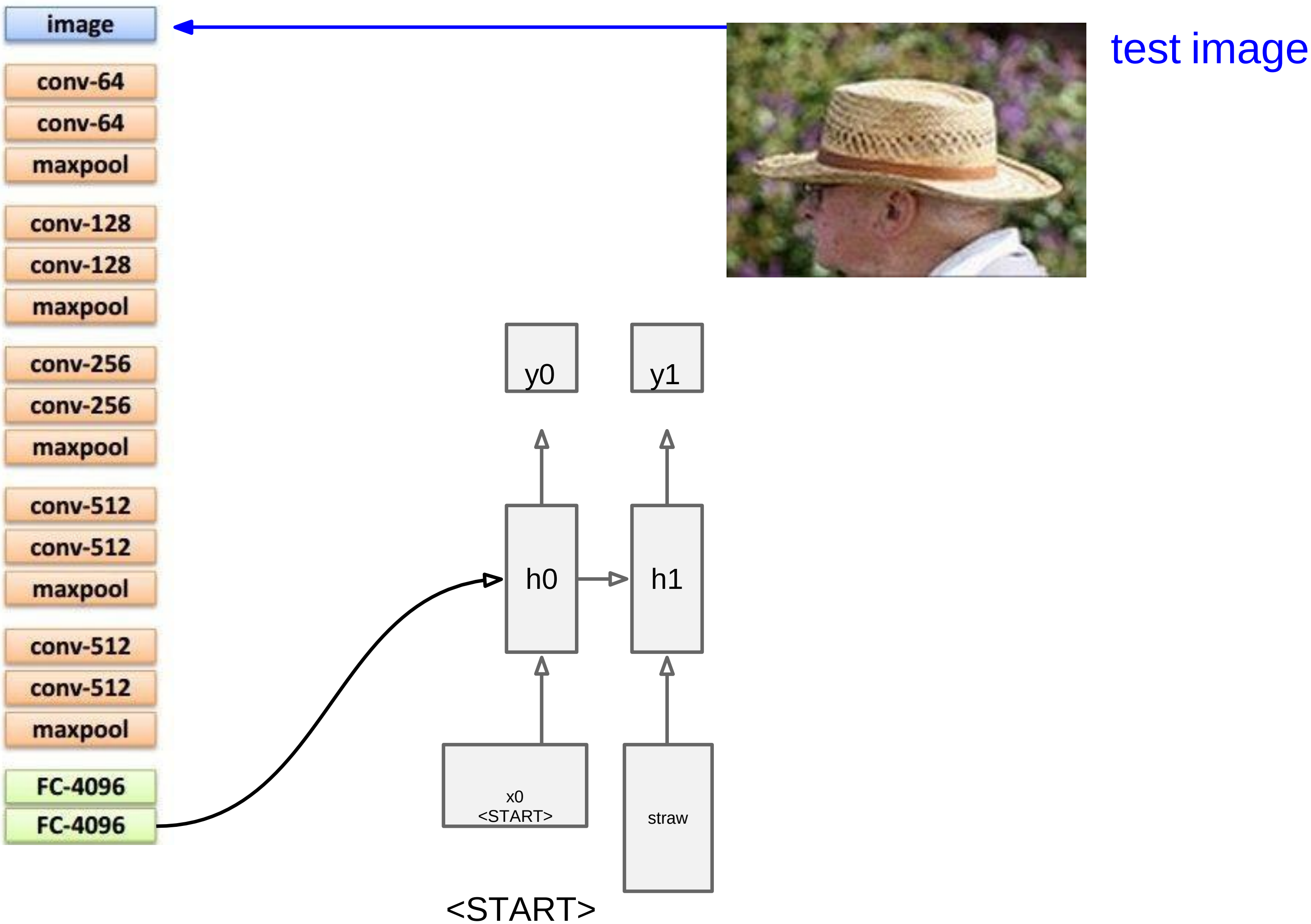
$$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * im)$$

Image Captioning



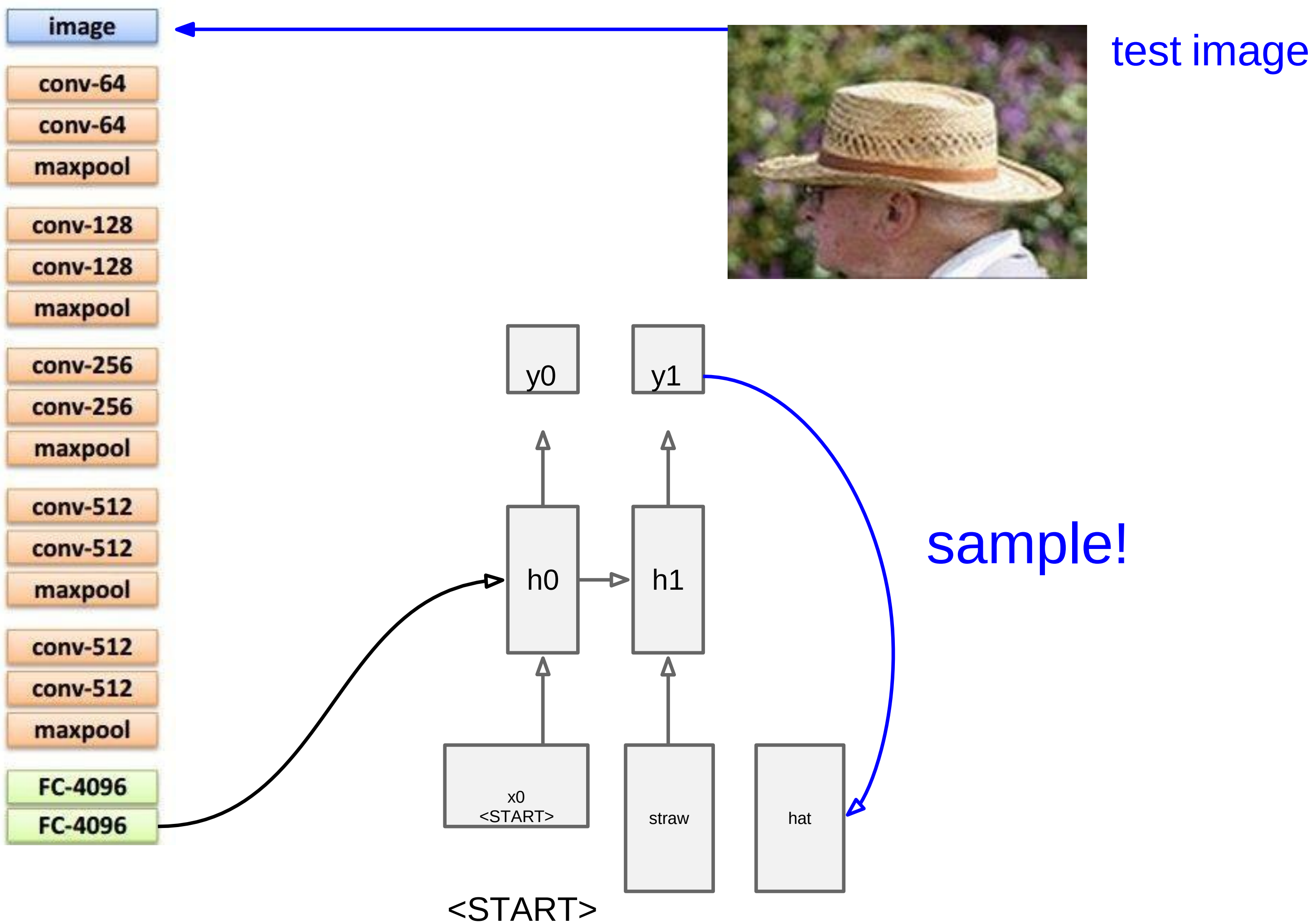
Andrej Karpathy

Image Captioning



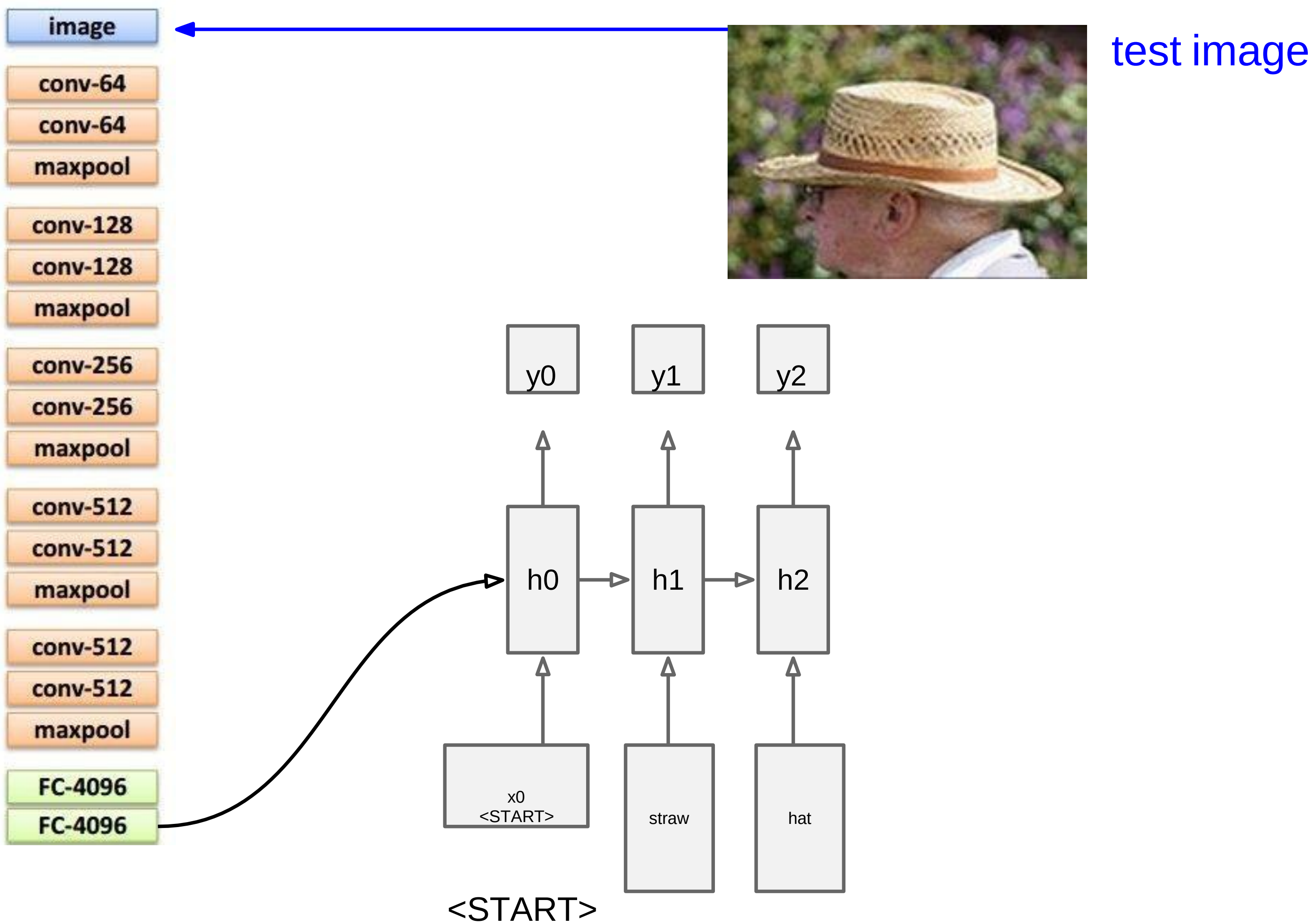
Andrej Karpathy

Image Captioning



Andrej Karpathy

Image Captioning

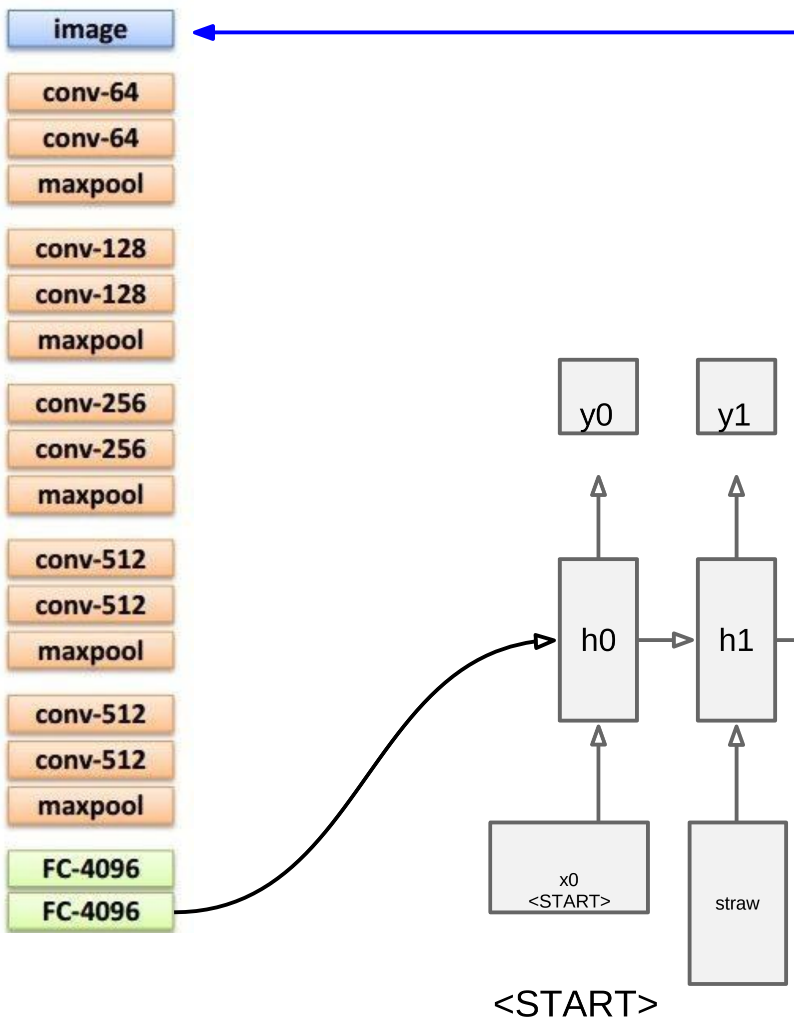


Andrej Karpathy

Image Captioning

test image

Caption generated:
“straw hat”



sample
<END> token
=> finish.

Adapted from Andrej Karpathy

Image Captioning

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"a young boy is holding a baseball bat."

"a cat is sitting on a couch with a remote control."

"a woman holding a teddy bear in front of a mirror."

"a horse is standing in the middle of a road."

Andrej Karpathy