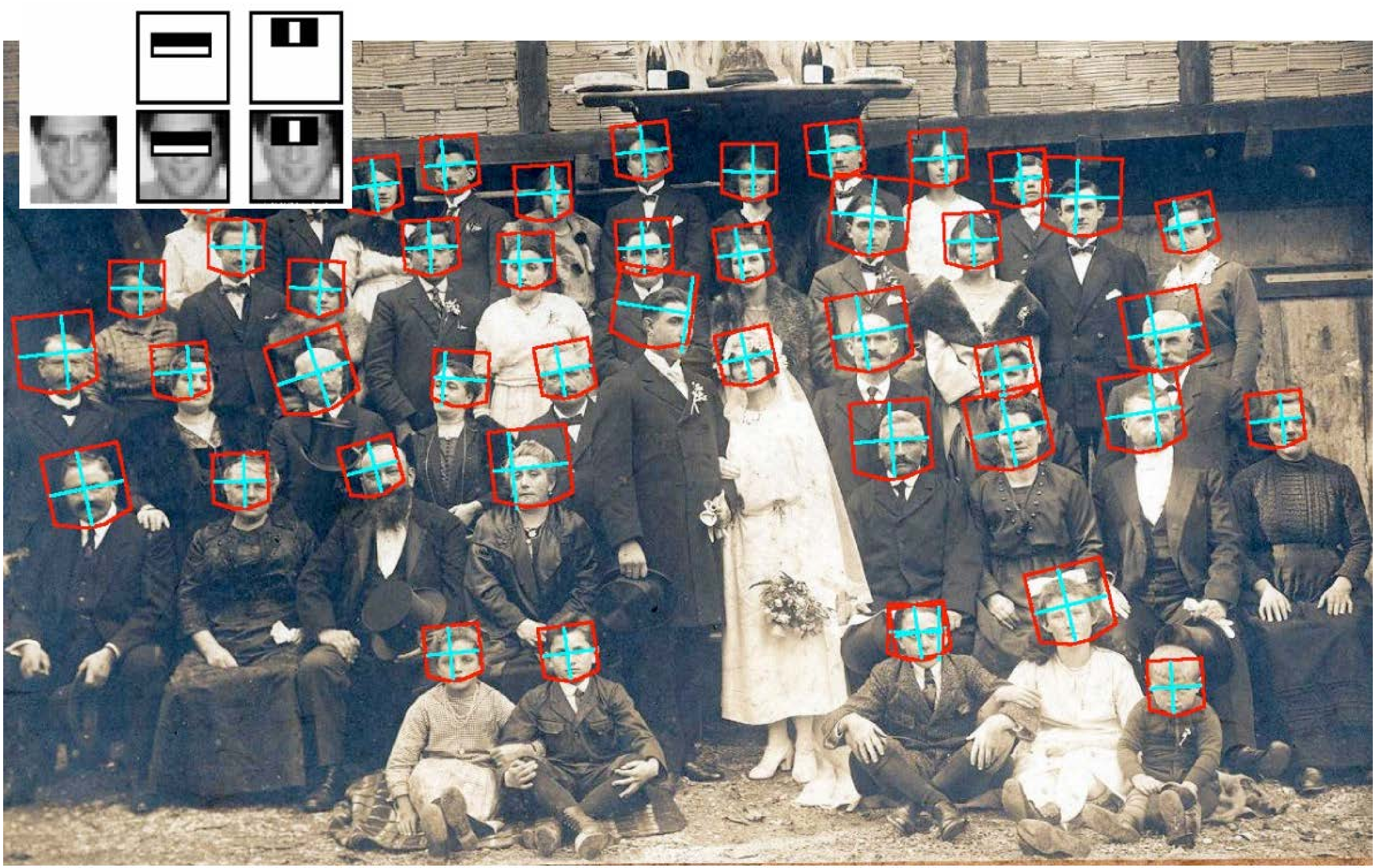
Deep Neural Networks Basics

For ECS 289G
Presented by Fanyi Xiao

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

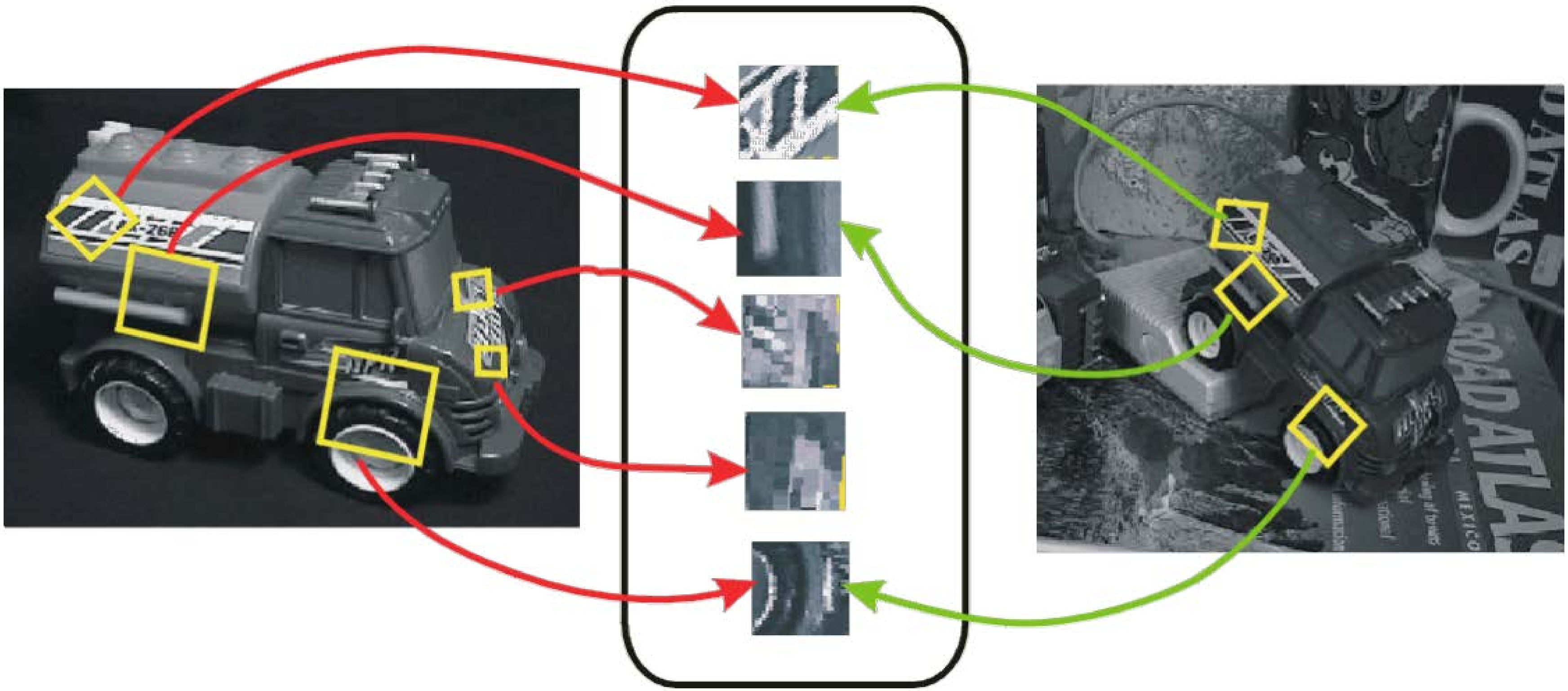
Computer Vision in the Pre-DNN Era

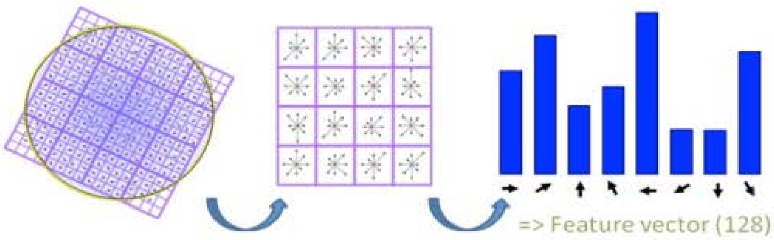


Face Detection, Viola & Jones, 2001

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Computer Vision in the Pre-DNN Era

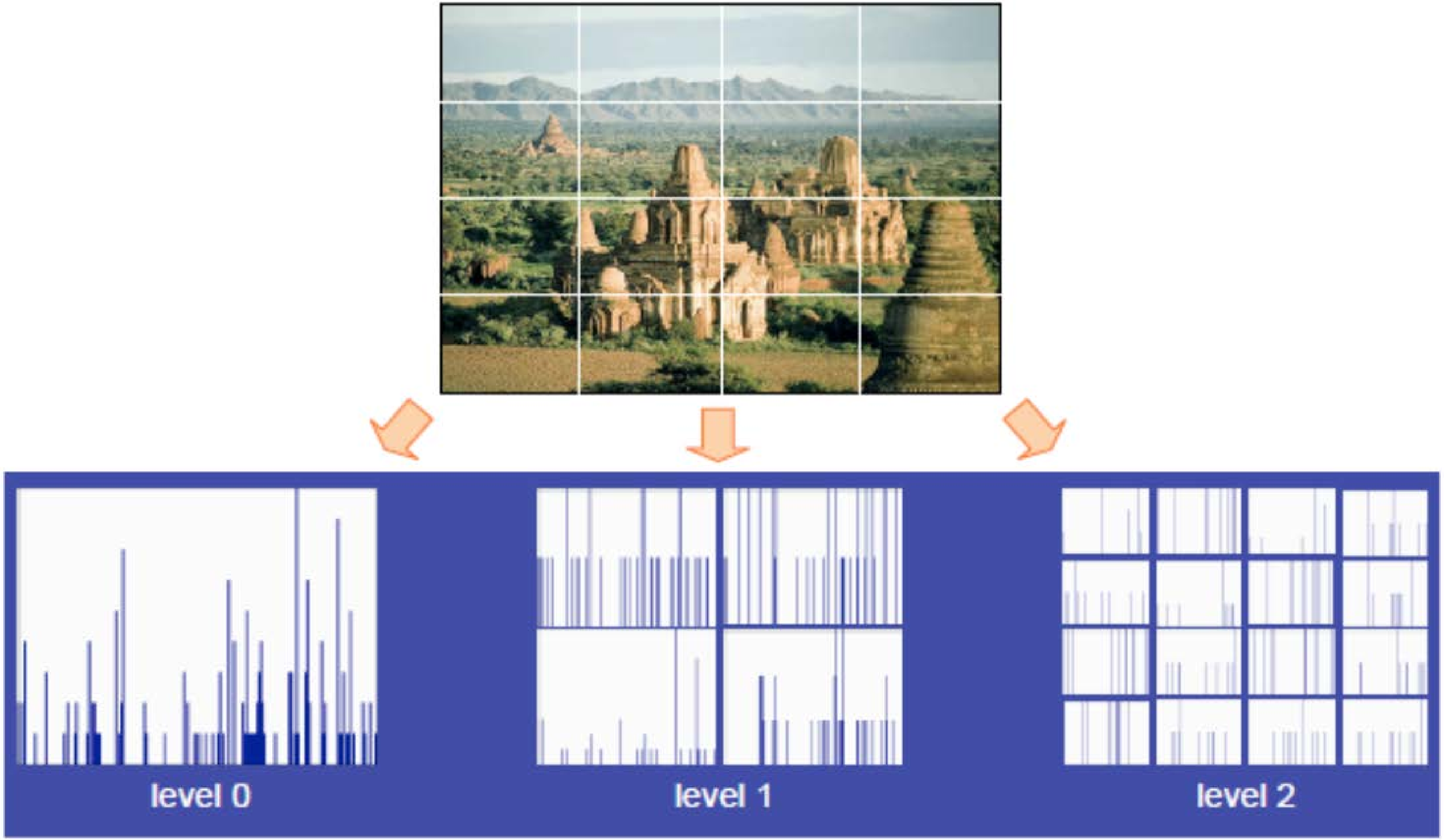




"SIFT" & Object Recognition, David Lowe, 1999

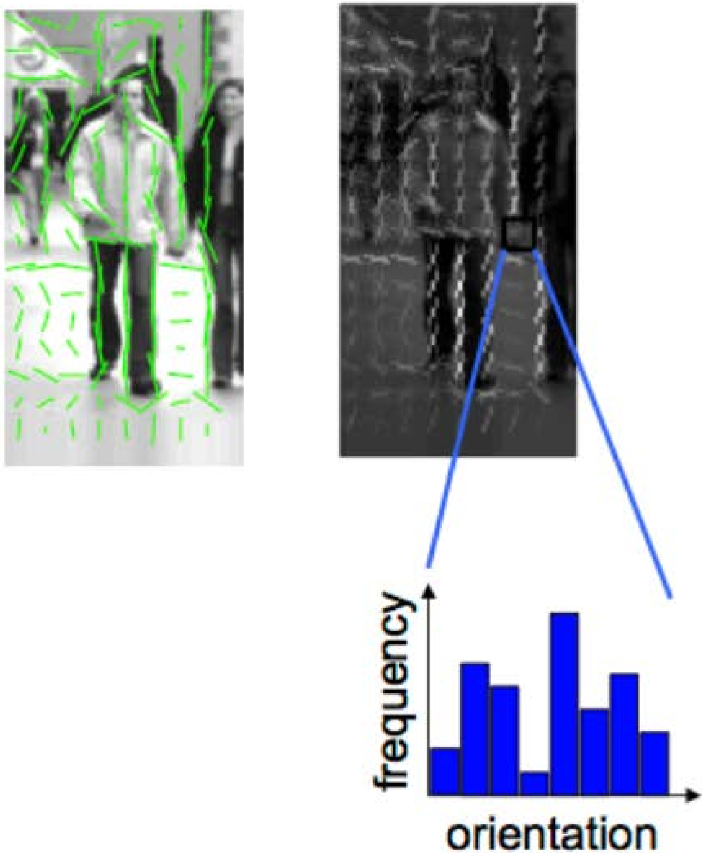
Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

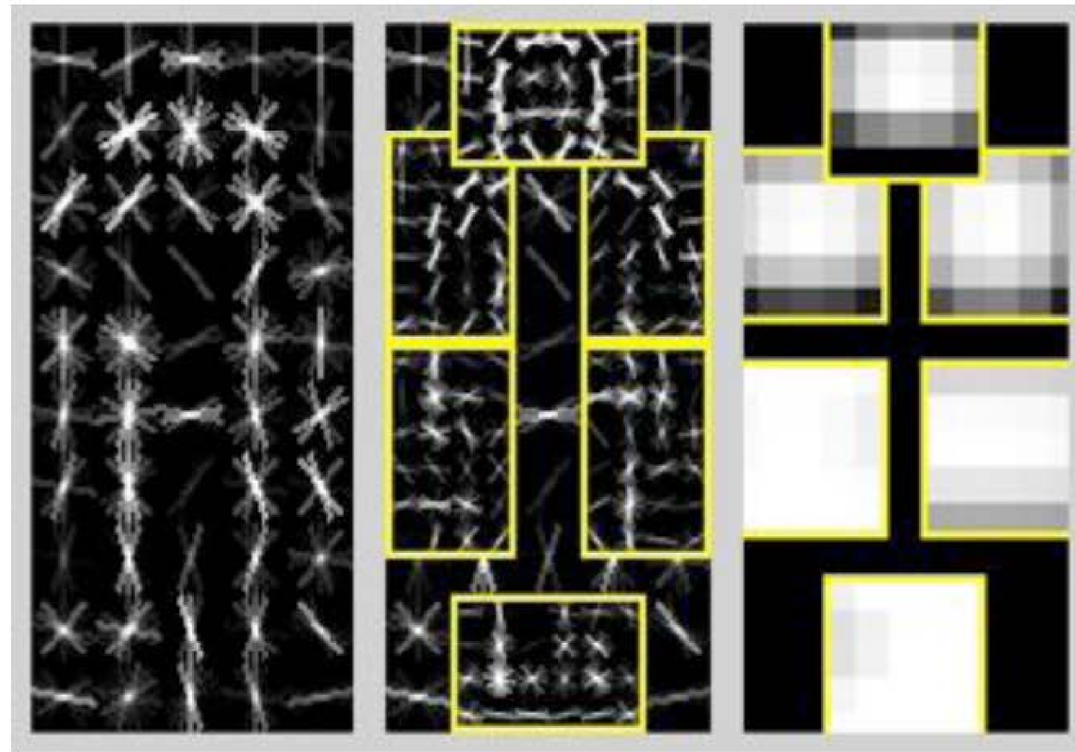
Computer Vision in the Pre-DNN Era



Spatial Pyramid Matching, Lazebnik, Schmid & Ponce, 2006

Computer Vision in the Pre-DNN Era



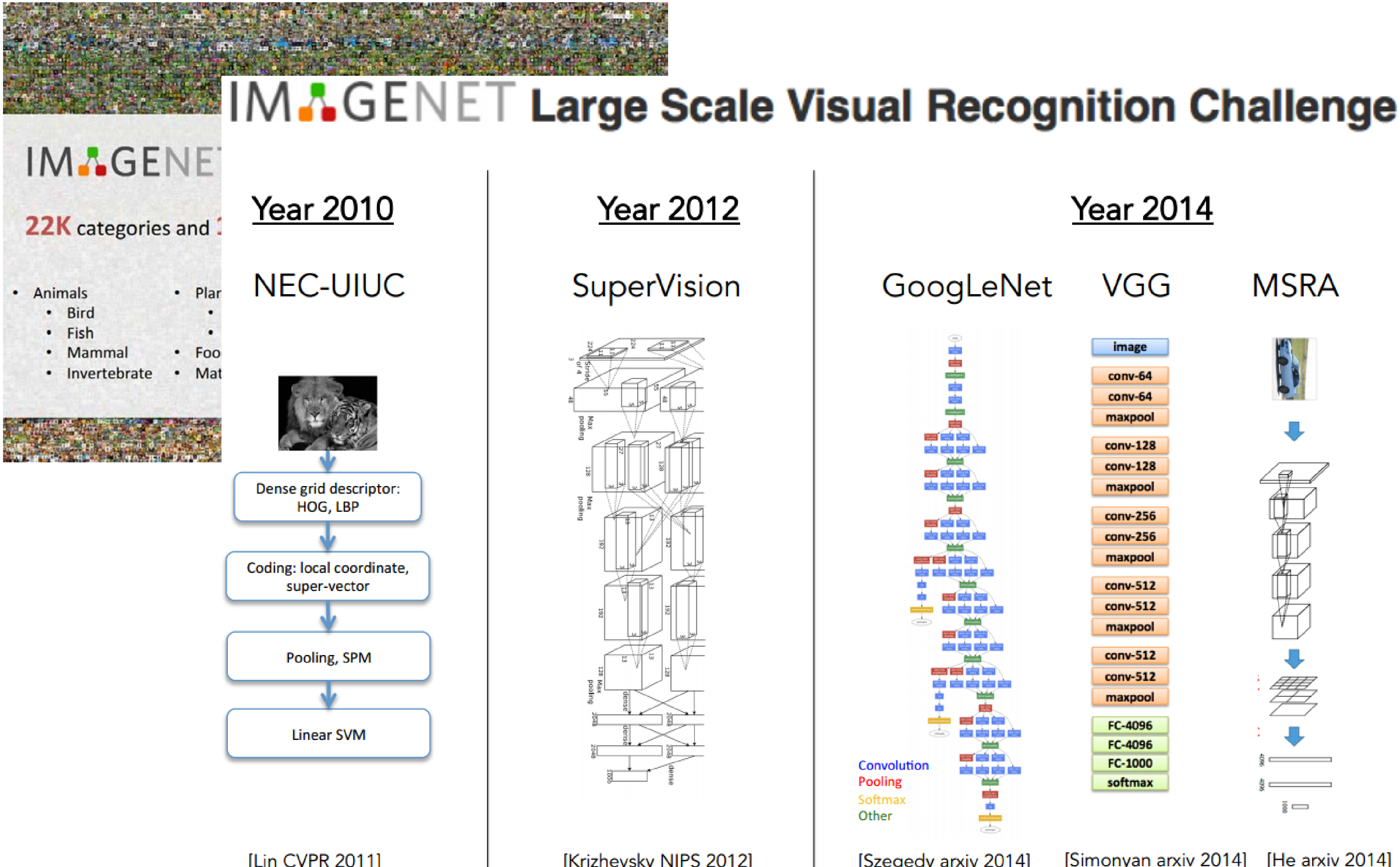


Histogram of Gradients (HoG)
Dalal & Triggs, 2005

Deformable Part Model
Felzenswalb, McAllester, Ramanan, 2009

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Emergence of DNNs in Vision



Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

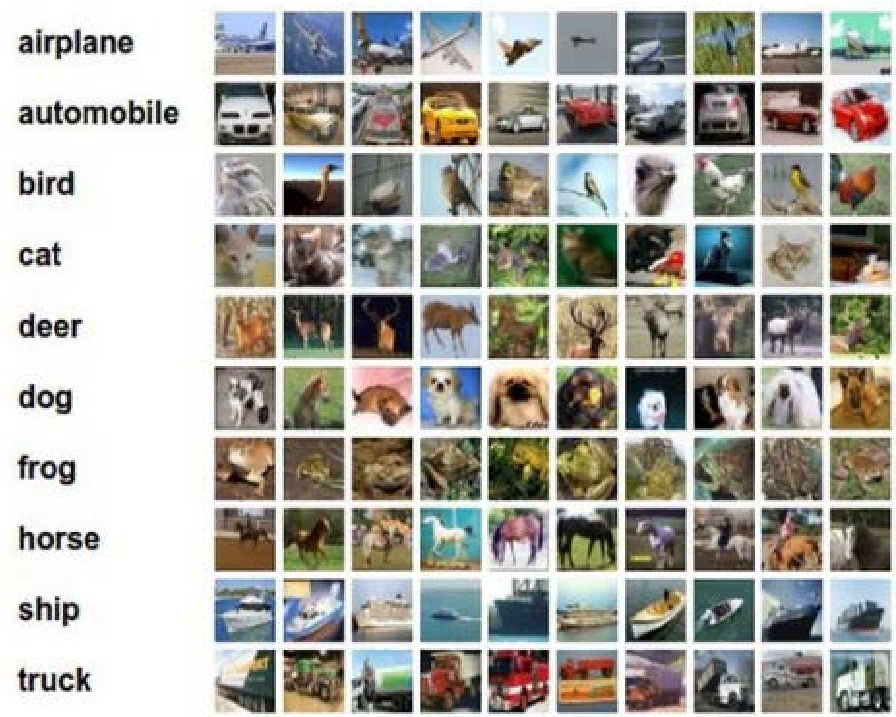
Neural Networks

Image Classification

assume given set of discrete labels
{dog, cat, truck, plane, ...}

cat

Data-driven approach



Learn visual features
"end-to-end"

Neural Networks

Compositional Models

Learned End-to-End

Hierarchy of Representations

- vision: pixel, motif, part, object
- text: character, word, clause, sentence
- speech: audio, band, phone, word

concrete → abstract

learning

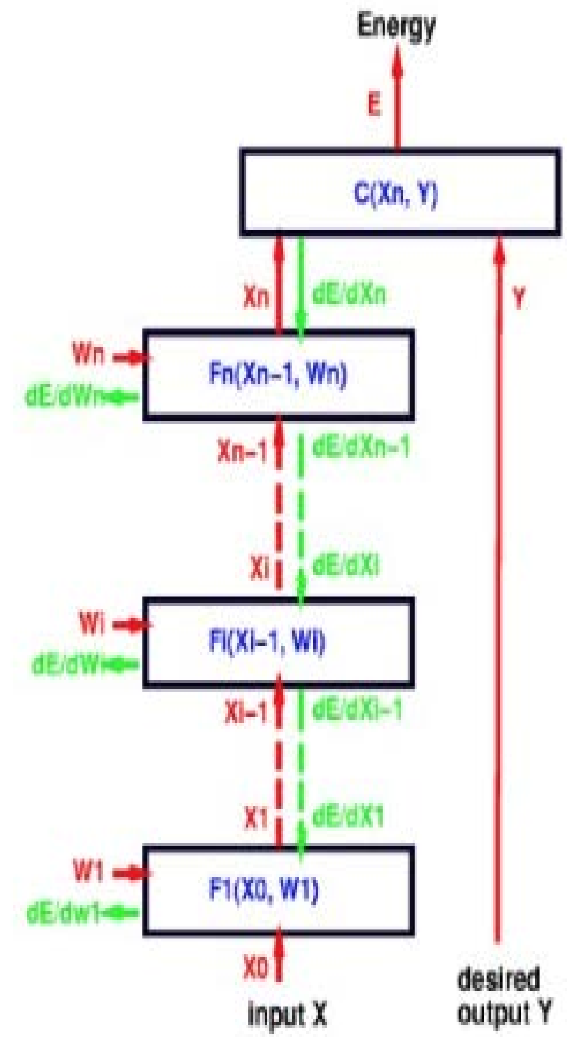


figure credit Yann LeCun, ICML '13 tutorial

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Neural Networks

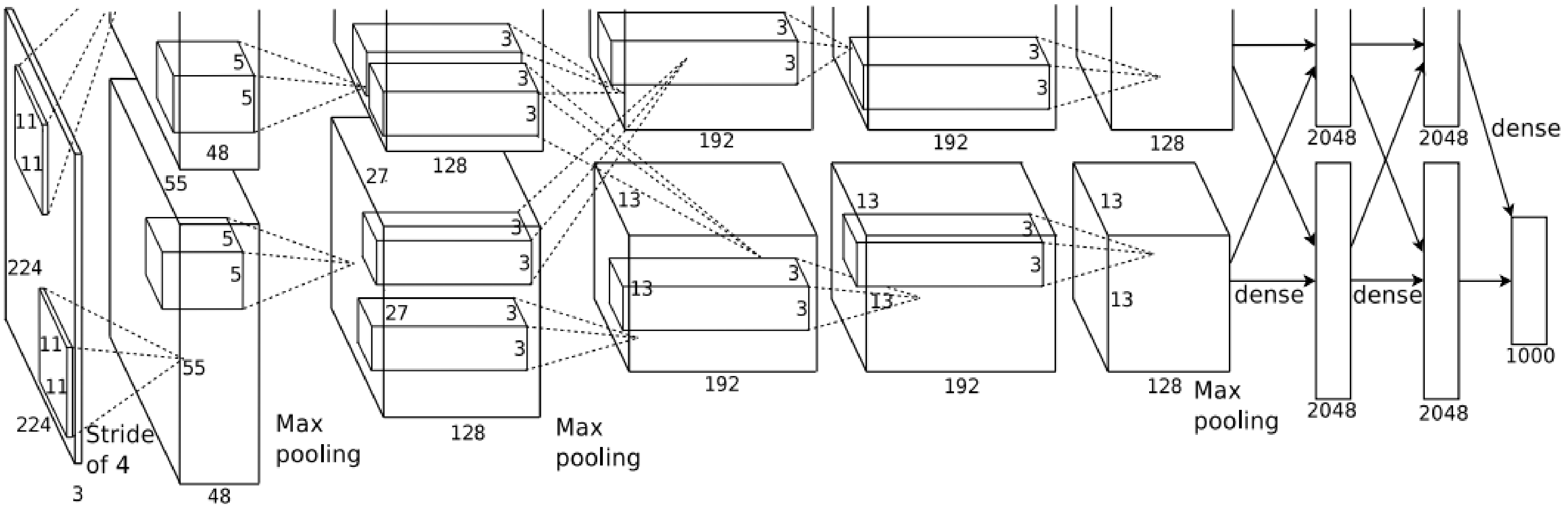
Three key ingredients for training an NN:

1. Score function
2. Loss function
3. Optimization

Neural Networks

Three key ingredients for training an NN:

1. Score function: y=f(x,W)



x -- 224*224*3 image patch
y -- 1000d vector

Neural Networks

Three key ingredients for training an NN:

2. Loss function: for example max-margin loss and cross-entropy loss

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Neural Networks

Three key ingredients for training an NN:

3. Optimization: simple gradient descent!

$$\frac{df(x)}{dx}$$

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Neural Networks

Three key ingredients for training an NN:

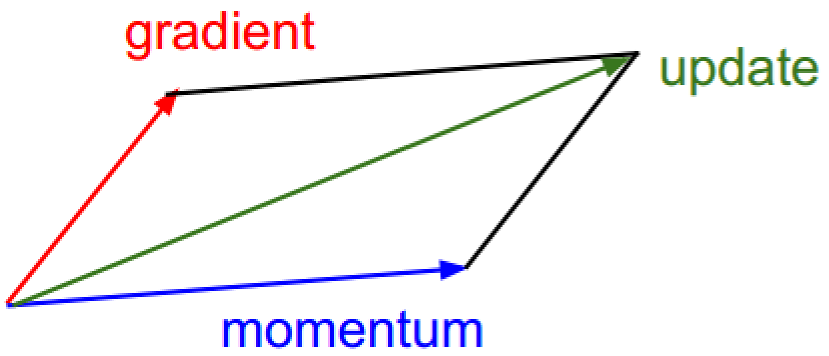
3. Optimization: in practice, *stochasitic (mini-batch) gradient descent*!

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Neural Networks

Three key ingredients for training an NN:

3. Optimization: in practice, stochasitic (mini-batch) gradient descent + *momentum*! (Many other optimization methods like adagrad/rmsprop)

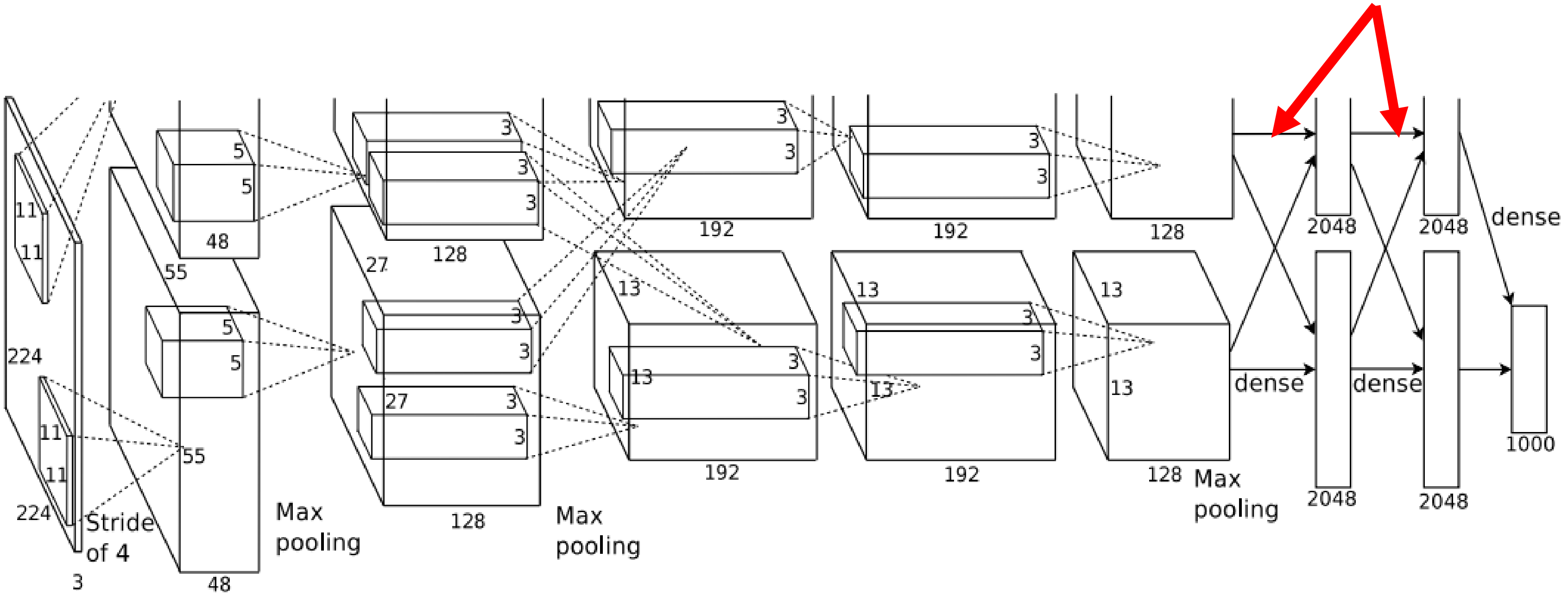



```
weights_grad = evaluate_gradient(loss_fun, data, weights)
vel = vel * 0.9 - step_size * weights_grad
weights += vel
```

Convolution Neural Networks

Let's take a closer look at AlexNet

Linear transformation:
y'=Wx+b

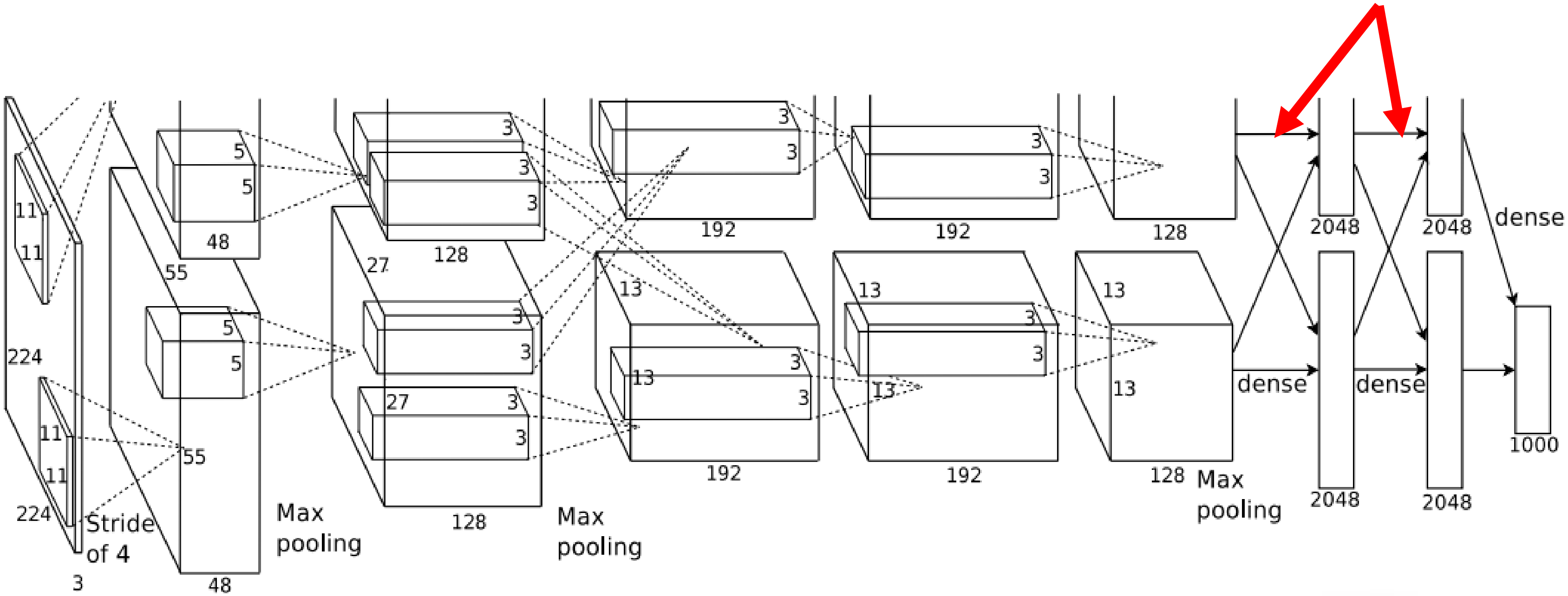


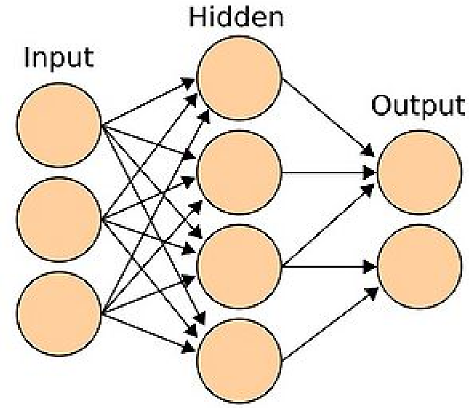
Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

Let's take a closer look at AlexNet

Linear transformation:
y'=Wx+b



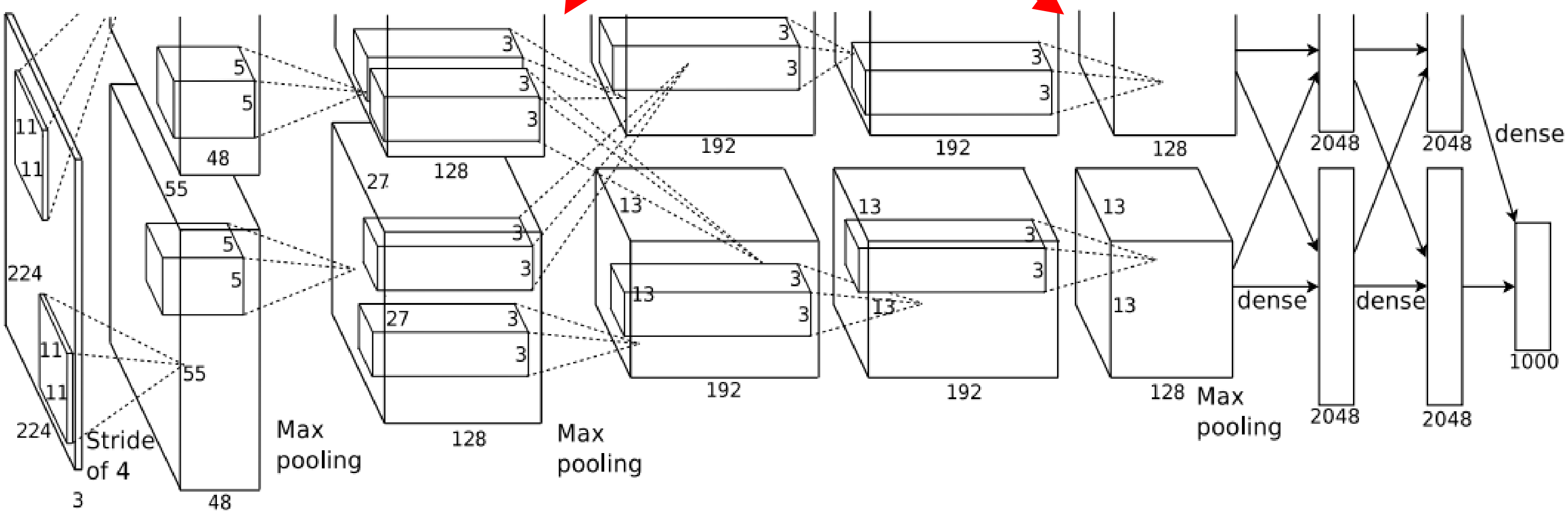


Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe

Convolution Neural Networks

Let's take a closer look at AlexNet

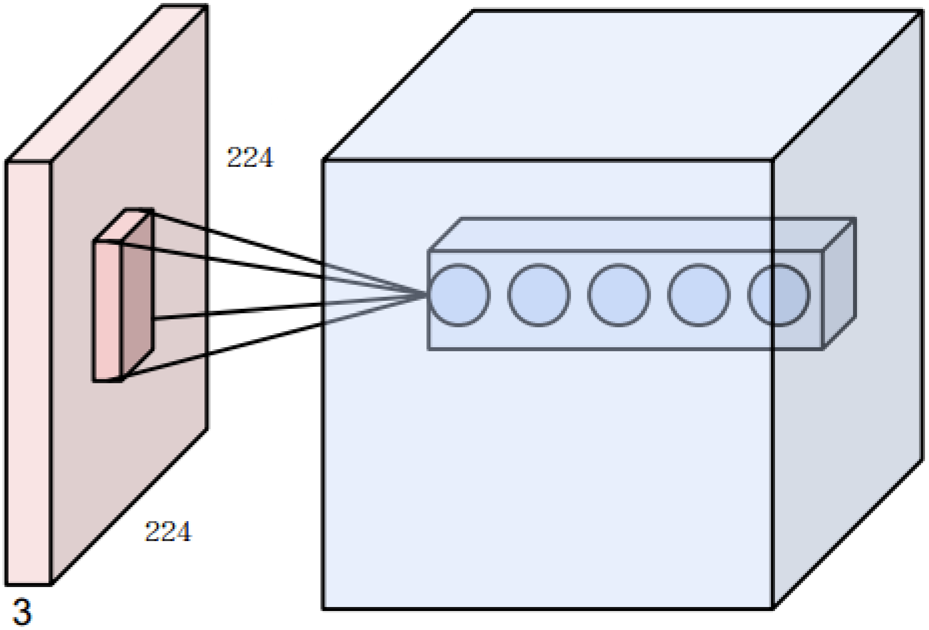
conv(h,w,stride)



Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

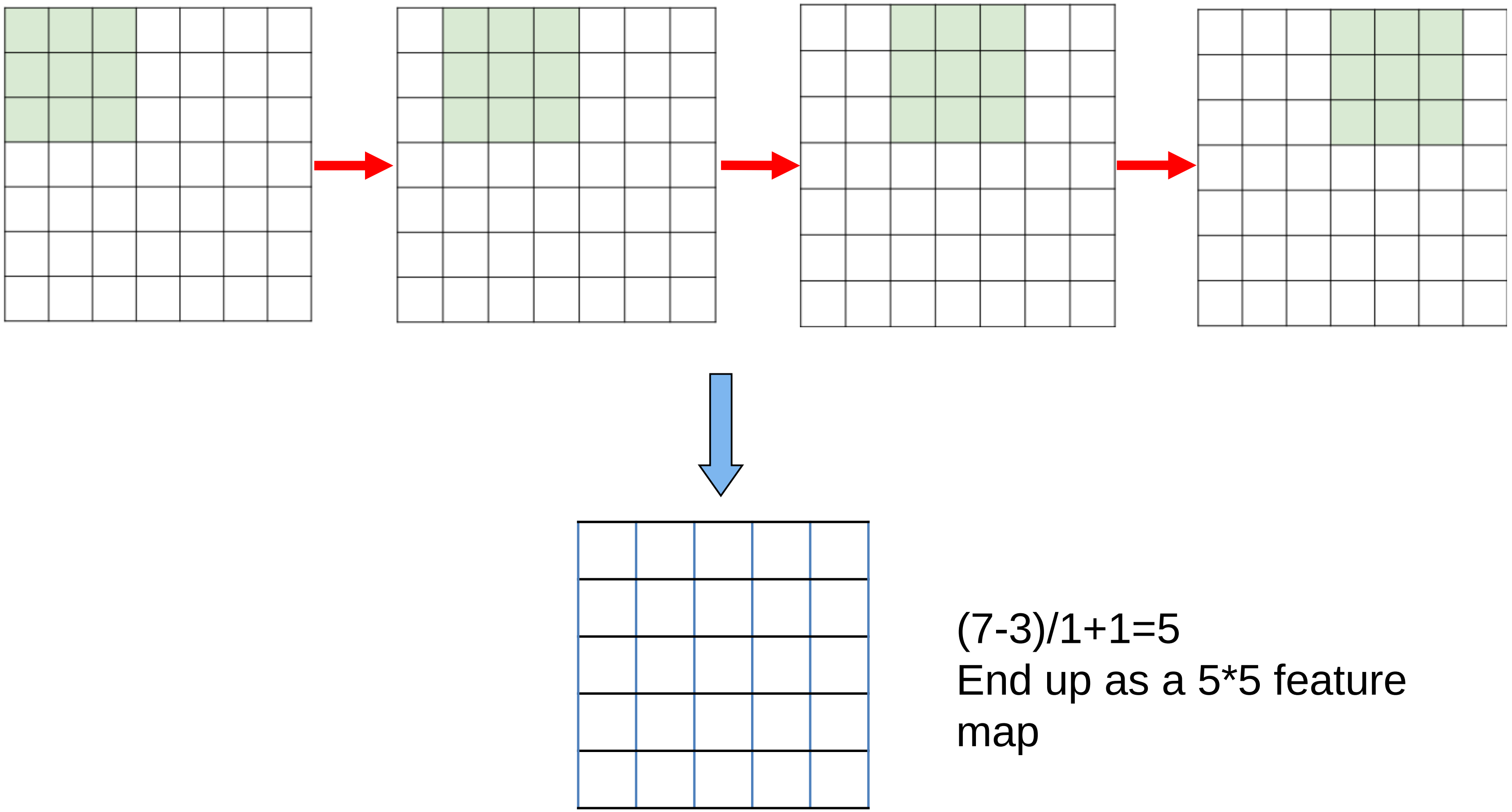
conv(h,w,stride)



Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

Example: conv(h=3,w=3,stride=1)



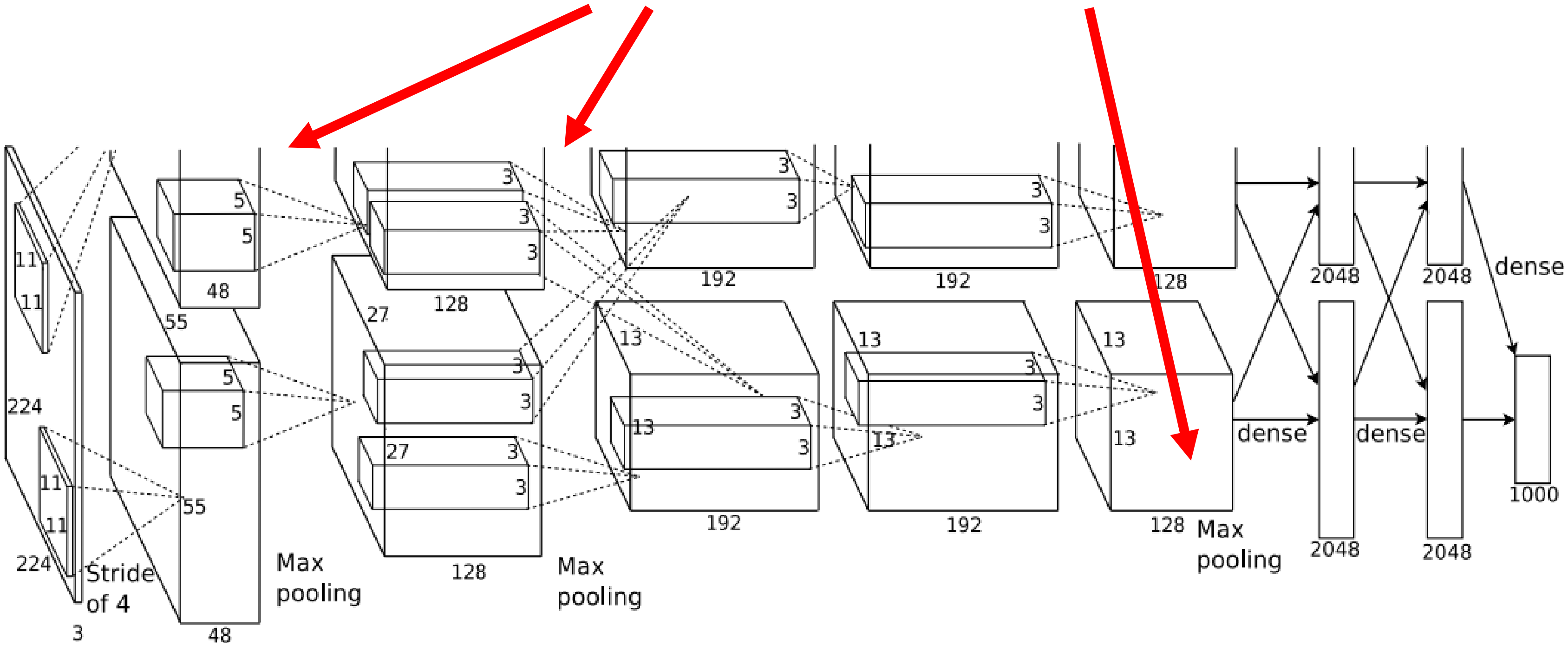
(7-3)/1+1=5
End up as a 5*5 feature map

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

Let's take a closer look at AlexNet

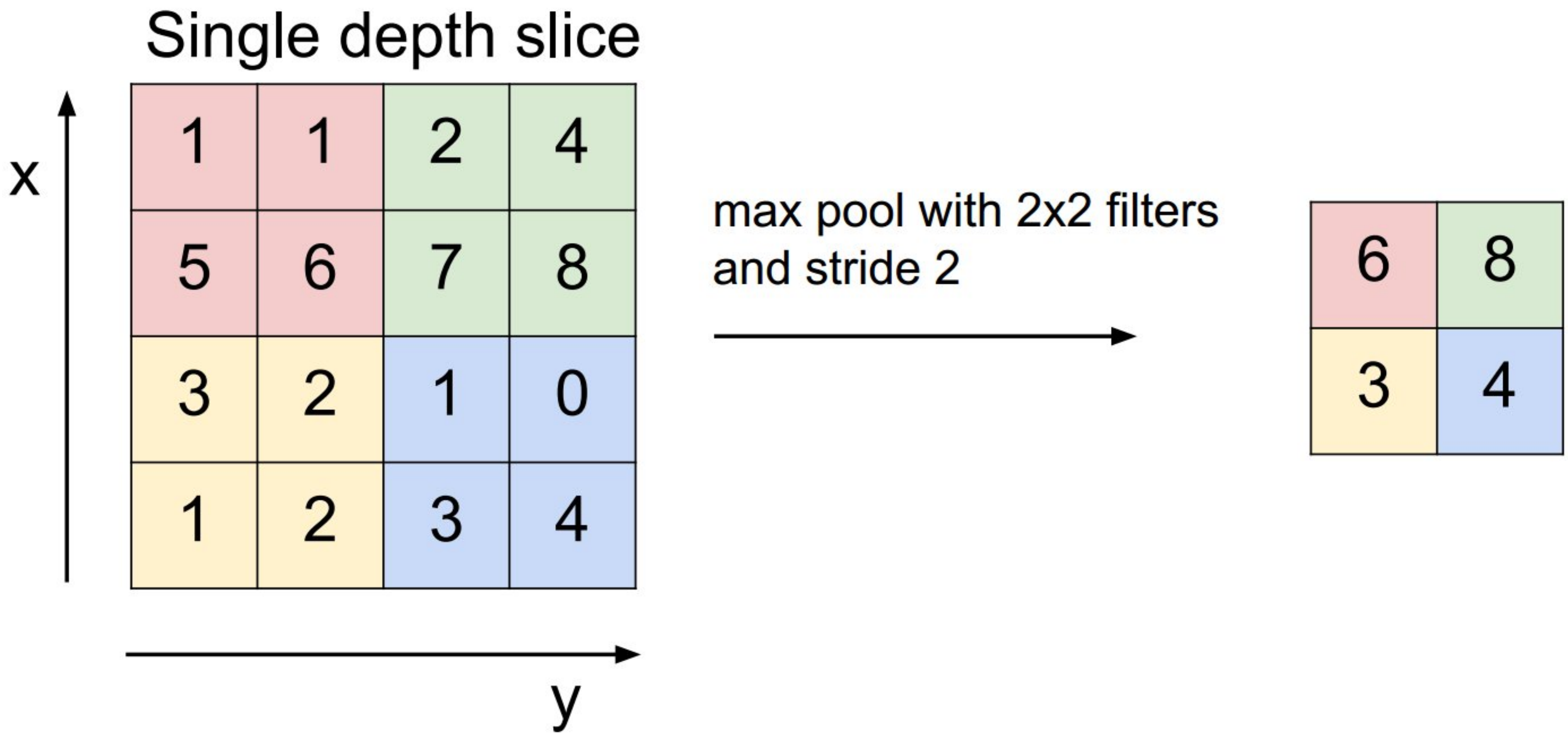
maxpool(h,w,stride)



Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

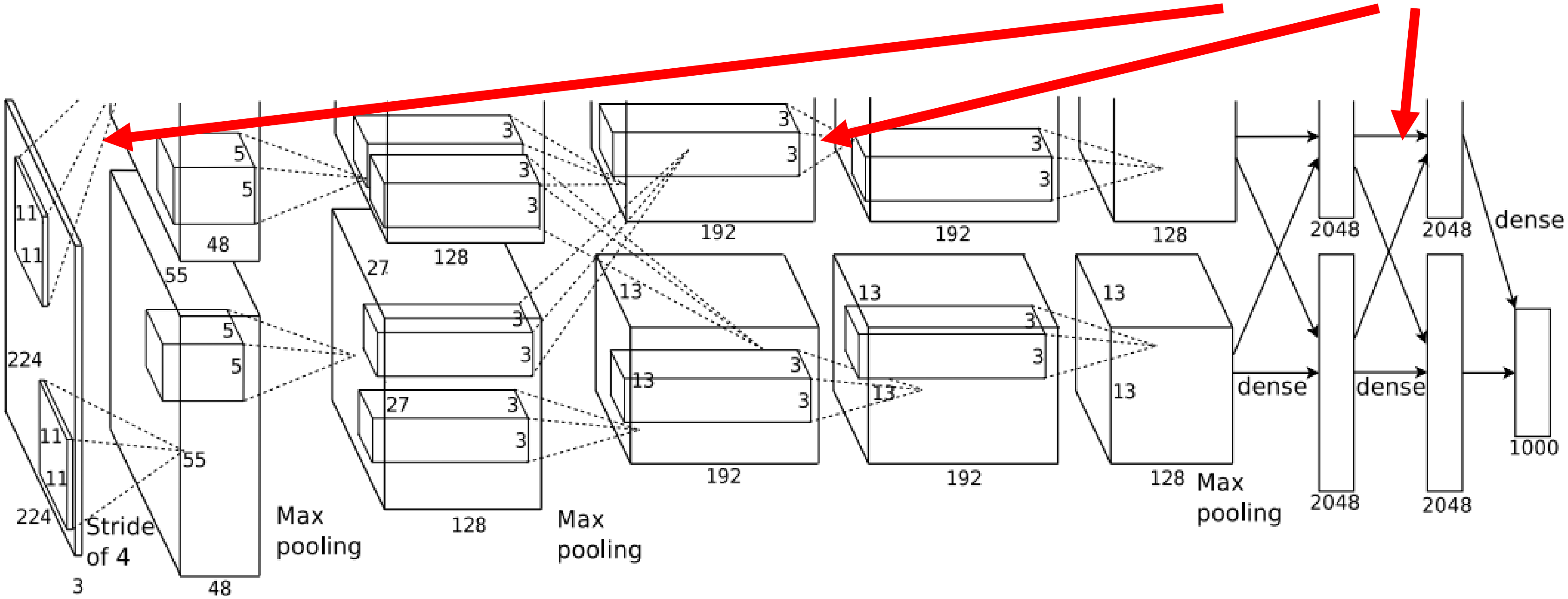
Example: maxpool(h=2,w=2,stride=2)



Convolution Neural Networks

Let's take a closer look at AlexNet

Relu: y=max(y',0)



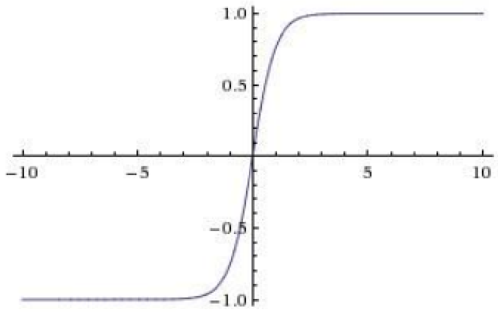
Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

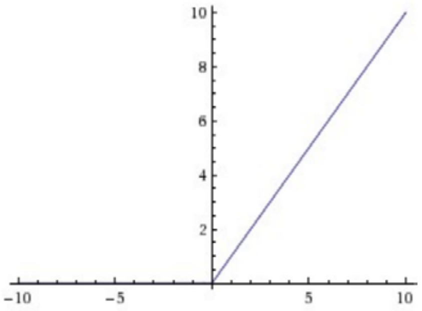
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$f(x) = \max(0,x)$$

Problems with tanh:
Saturated response



tanh(x)



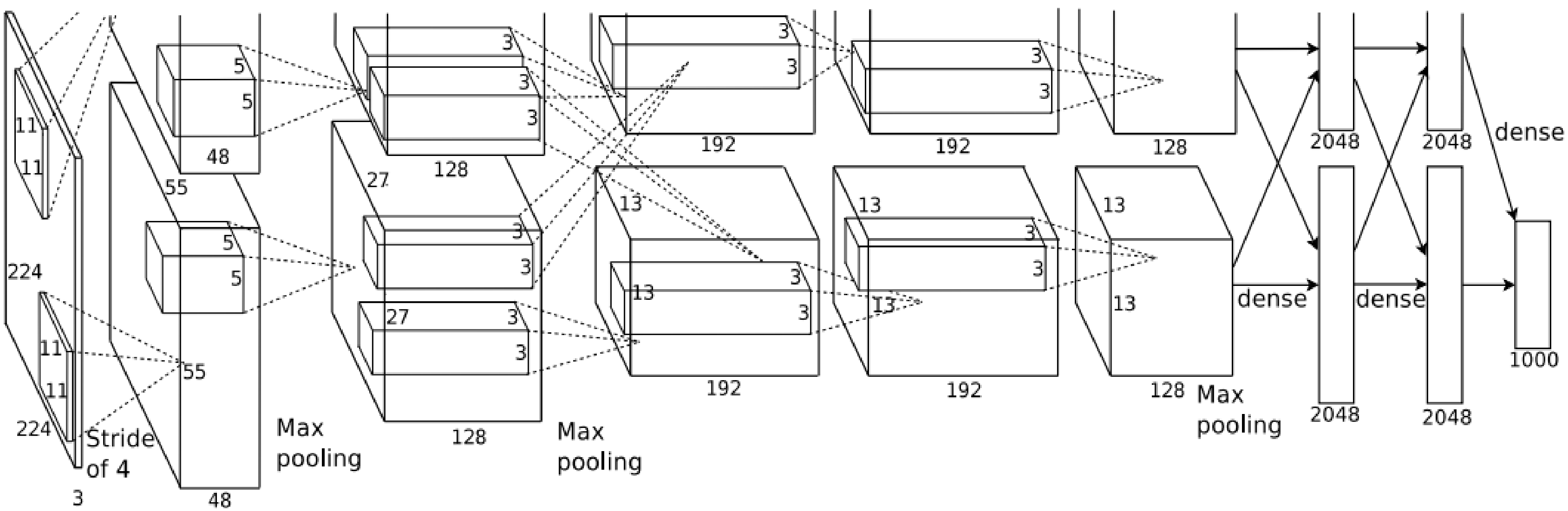
ReLU

Relu: y=max(y',0)
- Does not saturate
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice!

However, non-bounded response and dead when less than 0
(improved version leaky ReLU)

Convolution Neural Networks

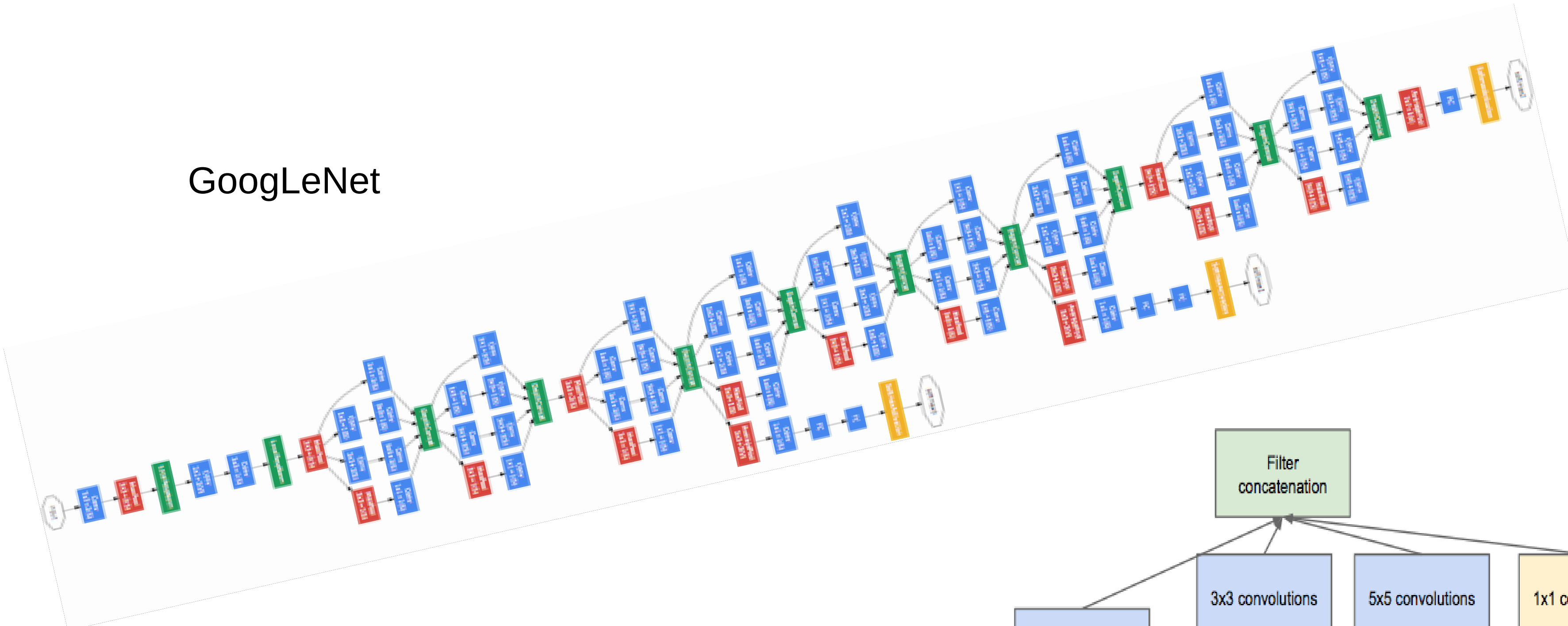
There are two key differences to Vanilla Neural Nets: neurons arranged in 3D volumes have local connectivity, share parameters



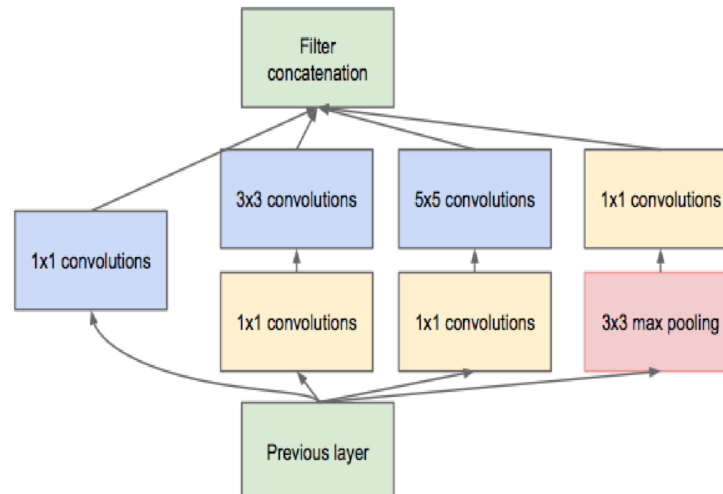
Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

GoogLeNet



ILSVRC14 Winners: **~6.6% Top-5 error**
- GoogLeNet: composition of multi-scale dimension-reduced modules (pictured)
- VGG: 16 layers of 3x3 convolution interleaved with max pooling + 3 fully-connected layers



+ depth
+ data
+ dimensionality reduction

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Convolution Neural Networks

Object Detection

R-CNN: Region-based Convolutional Networks

http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/detection.ipynb

Full R-CNN scripts available at

https://github.com/rbgirshick/rcnn

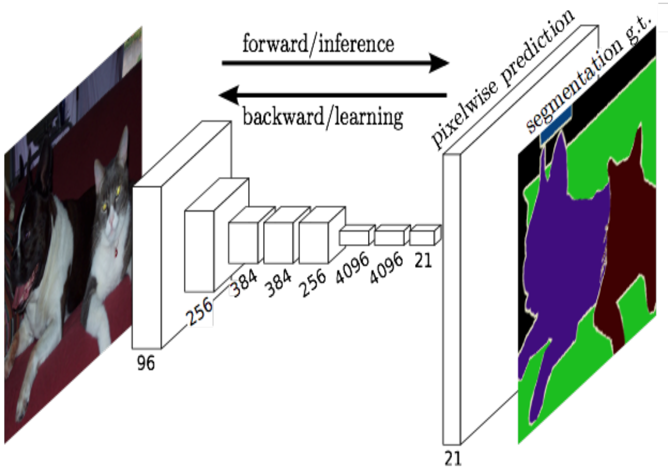
Ross Girshick et al.
Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR14.

Fast R-CNN

arXiv and code

Segmentation

Fully convolutional networks for pixel prediction
applied to semantic segmentation
end-to-end learning
efficiency in inference and learning
175 ms per-image prediction
multi-modal, multi-task

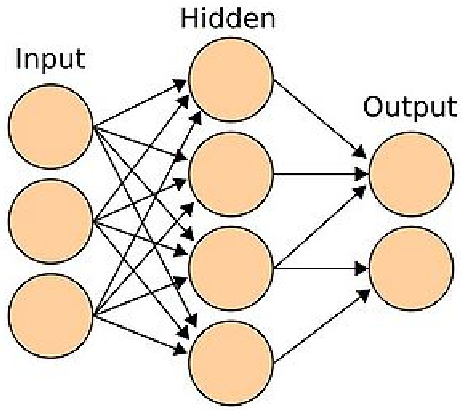


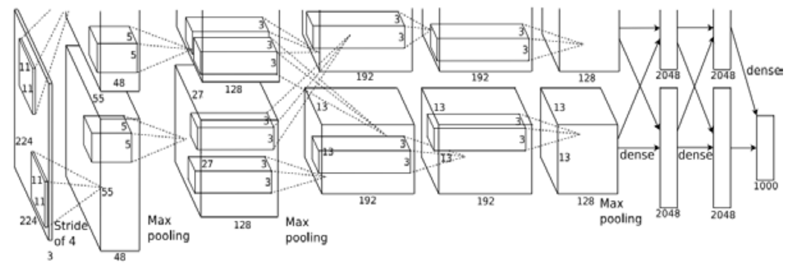
Further applications
- depth estimation
- denoising

arXiv and pre-release

Jon Long* & Evan Shelhamer*,

Problem with Feed-forward Nets

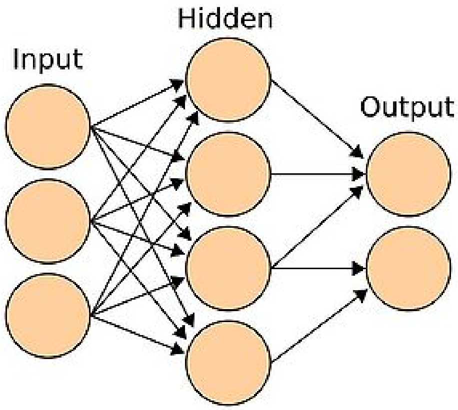


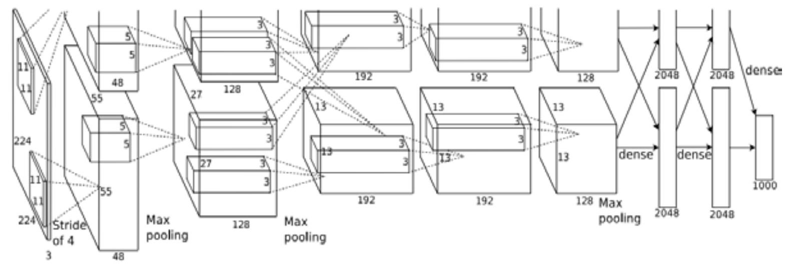


What if we want to be able to have a model telling us what's the probability of the following two sententes, resepctively:

1. The cat sat on the mat
2. The mat is having dinner with the cat

Problem with Feed-forward Nets





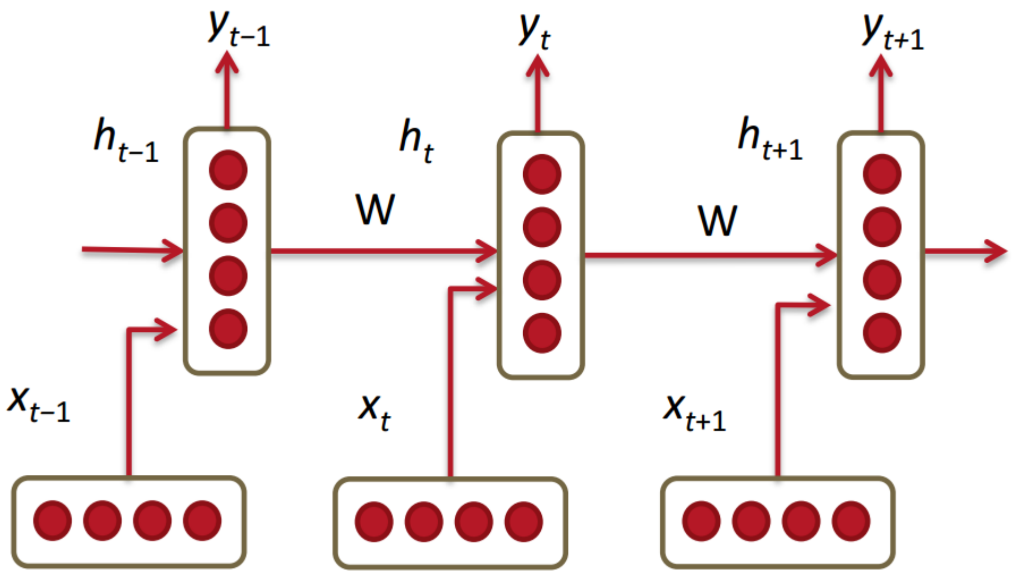
What if we want to be able to have a model telling us what's the probability of the following two sententes, resepctively:

1. The cat sat on the mat
2. The mat is having dinner with the cat

Cannot handle variable length input

Recurrent Neural Net

RNNs tie the weights at each time step



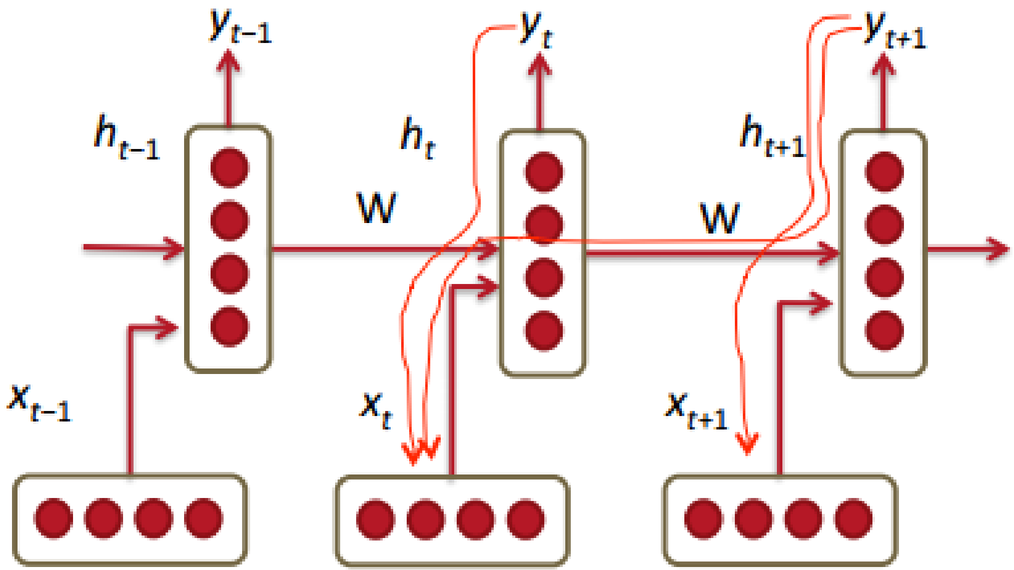
$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)}h_t\right)$$

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Recurrent Neural Net

Training of RNNs is hard...



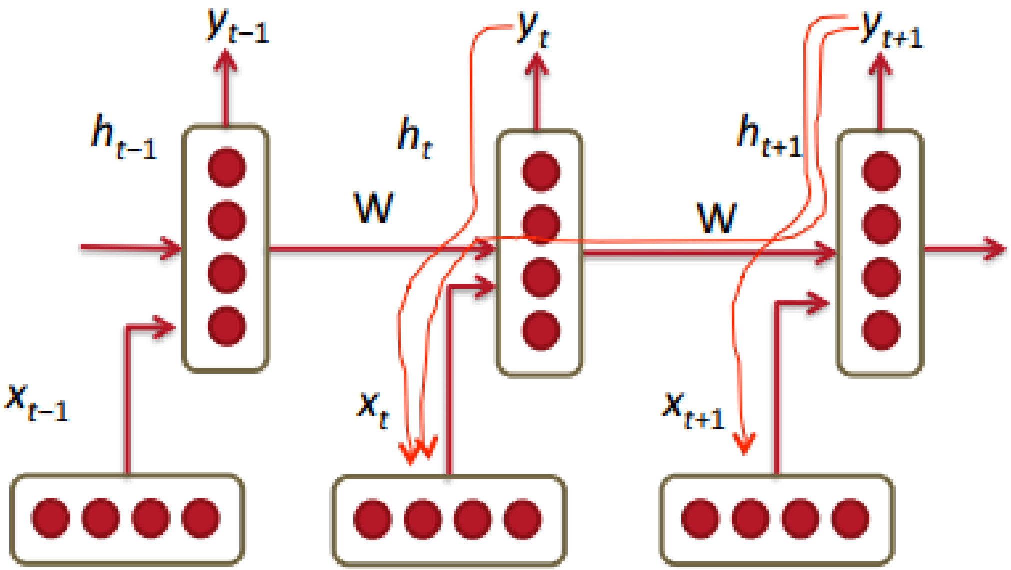
$$h_t = W f(h_{t-1}) + W^{(hx)} x_{[t]}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^T \mathrm{diag}[f'(h_{j-1})]$$

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Recurrent Neural Net

Training of RNNs is hard...



Solution 1: clip the gradient!

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

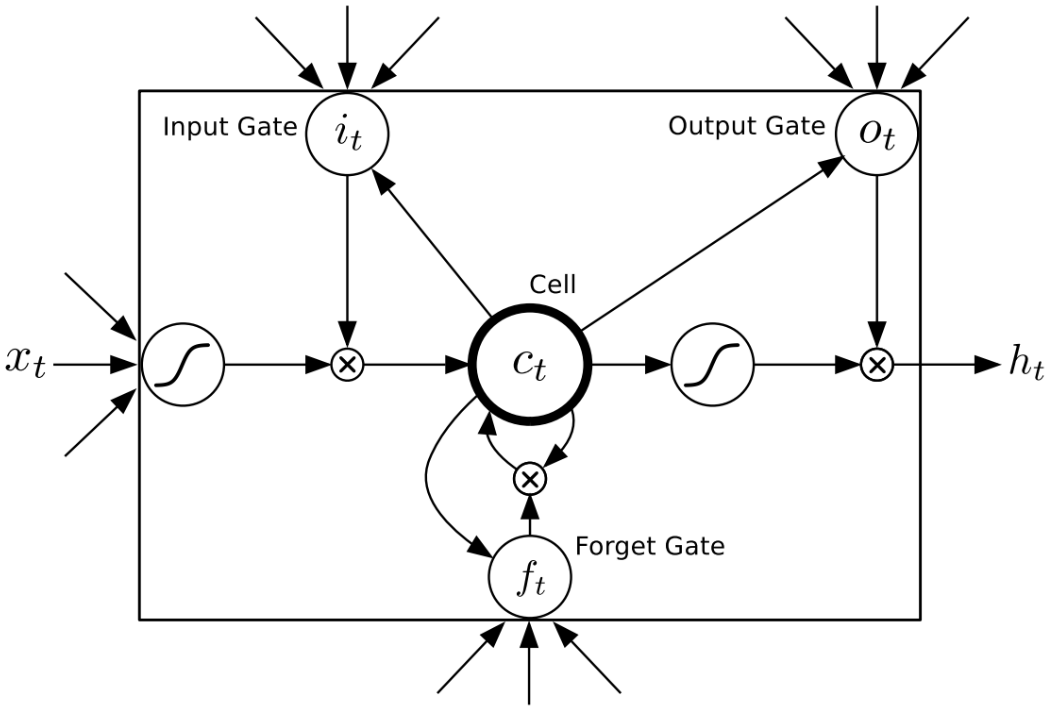
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Some theory: On the difficulty of training recurrent neural networks, Pascanu et al. ICML2013

Recurrent Neural Net

Training of RNNs is hard...

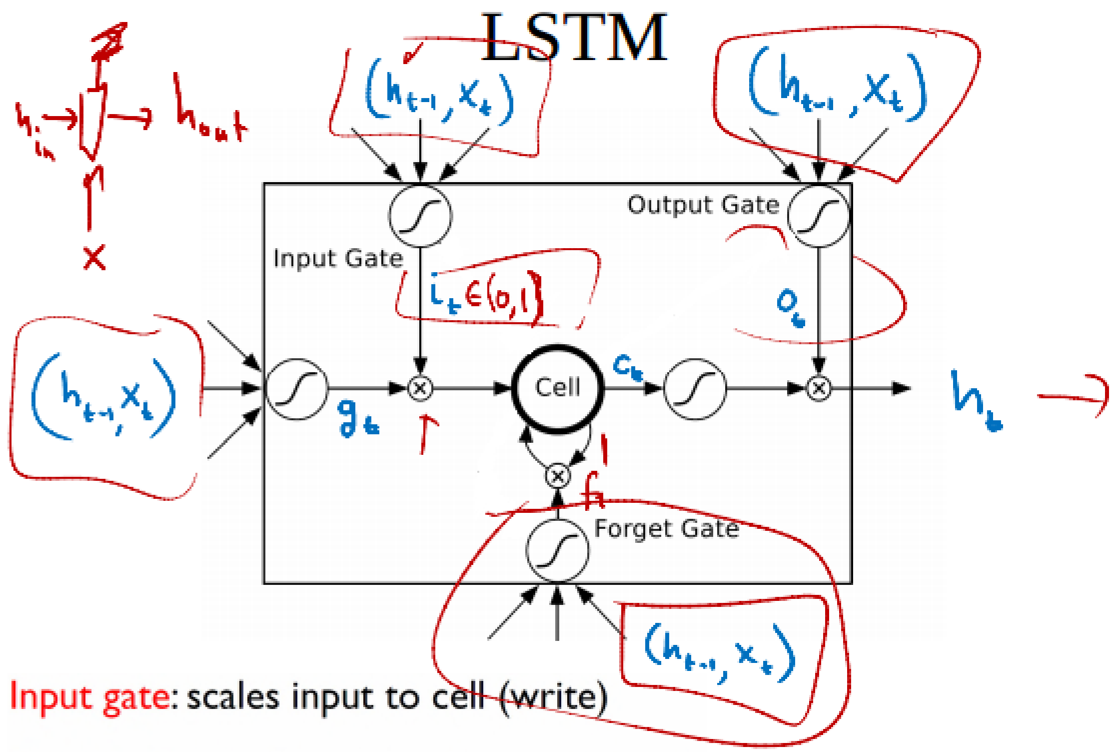
Solution 2: NNs with gating units (LSTM/GRU)



Recurrent Neural Net

Training of RNNs is hard...

Solution 2: nets with gating units (LSTM/GRU)



Input gate: scales input to cell (write)

Output gate: scales output from cell (read)

Forget gate: scales old cell value (reset)

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Recurrent Neural Net

Training of RNNs is hard...

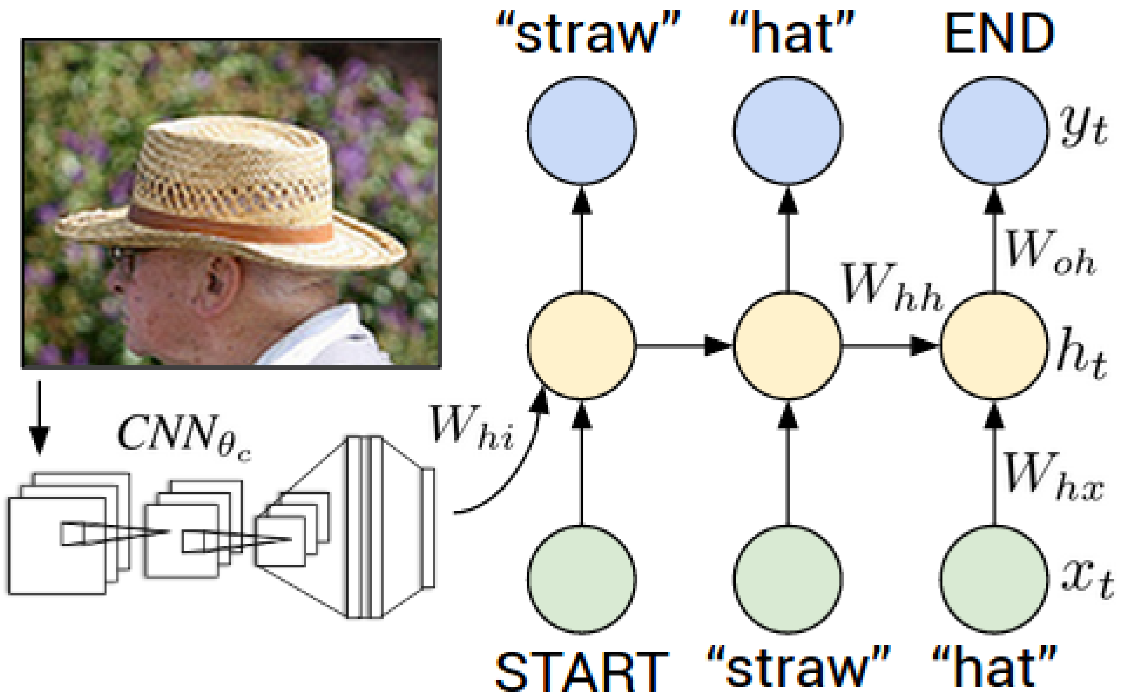
Solution 2: nets with gating units (LSTM/GRU)

$$\mathbf{i}_t = Sigm(\boldsymbol{\theta}_{xi}\mathbf{x}_t + \boldsymbol{\theta}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{f}_t = Sigm(\boldsymbol{\theta}_{xf}\mathbf{x}_t + \boldsymbol{\theta}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{o}_t = Sigm(\boldsymbol{\theta}_{xo}\mathbf{x}_t + \boldsymbol{\theta}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)$$
$$\mathbf{g}_t = Tanh(\boldsymbol{\theta}_{xg}\mathbf{x}_t + \boldsymbol{\theta}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} \boxed{+} \mathbf{i}_t \odot \mathbf{g}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot Tanh(\mathbf{c}_t)$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \odot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 y_1 \\ x_2 y_2 \end{bmatrix}$$

RNN in vision

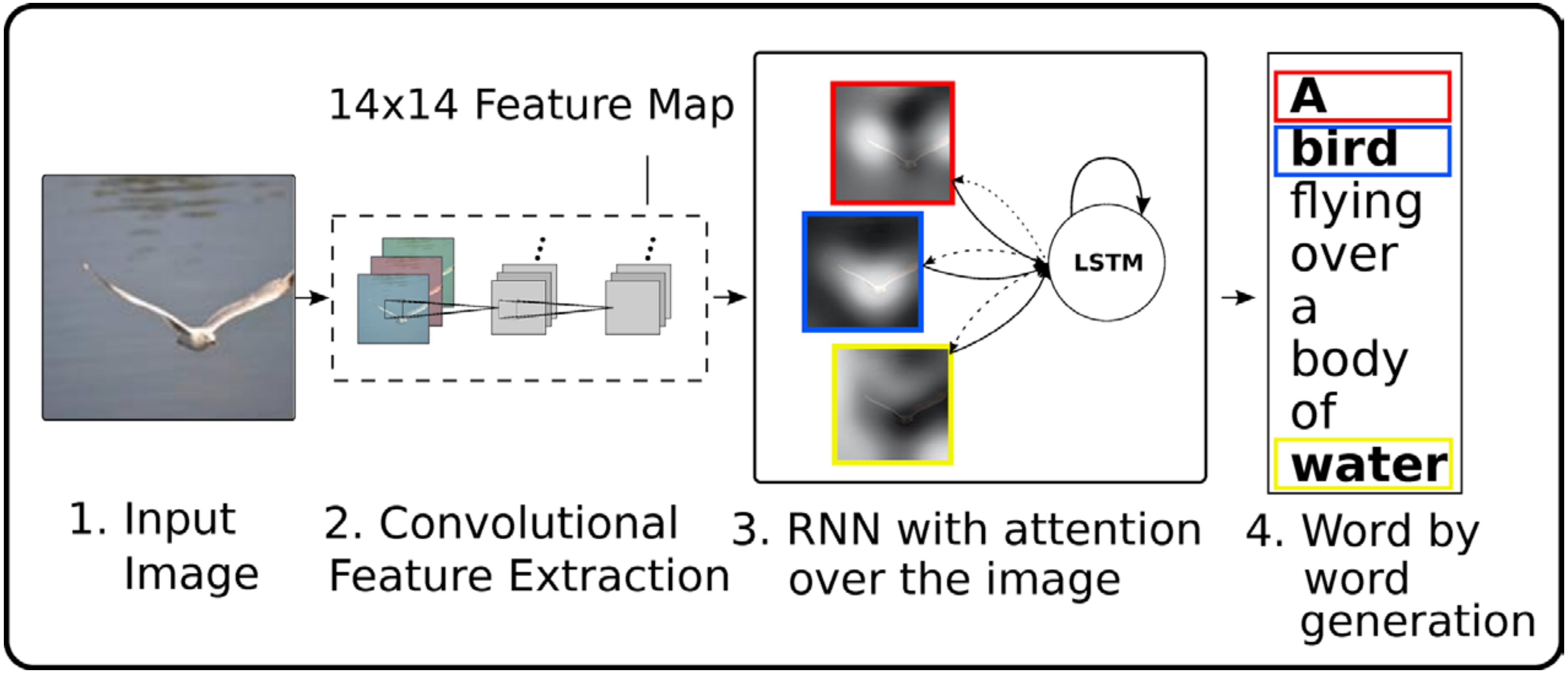
Image captioning



Deep Visual-Semantic Alignments for Generating Image Descriptions, Andrej Karpathy et al.

RNN in vision

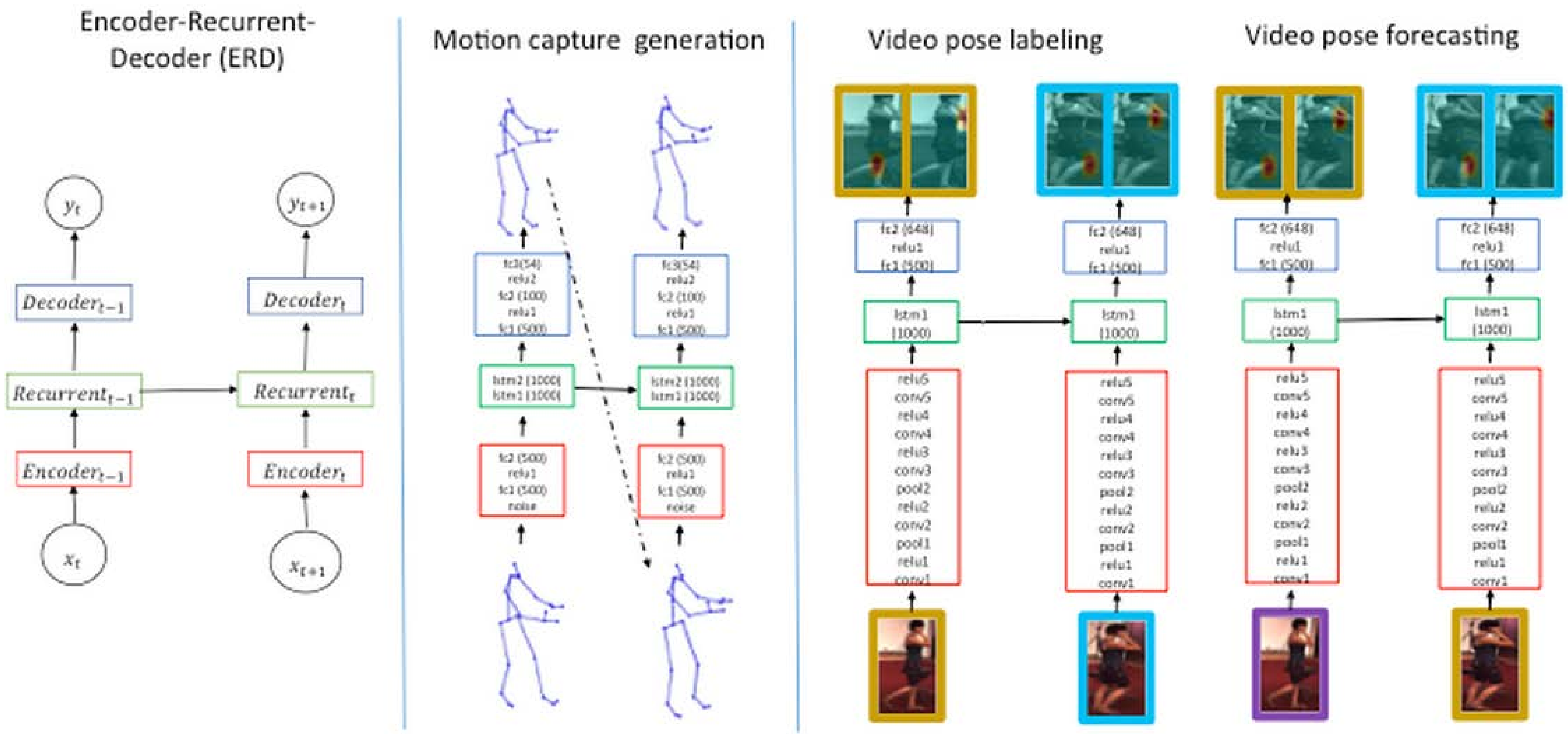
Visual attention model



Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, Kelvin Xu et al.

RNN in vision

RNNs for Human Dynamics



Recurrent Network Models for Human Dynamics, Katerina Fragkiadaki et al.

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

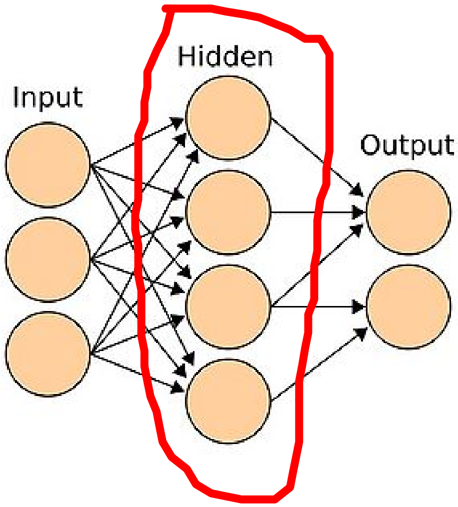
Tricks

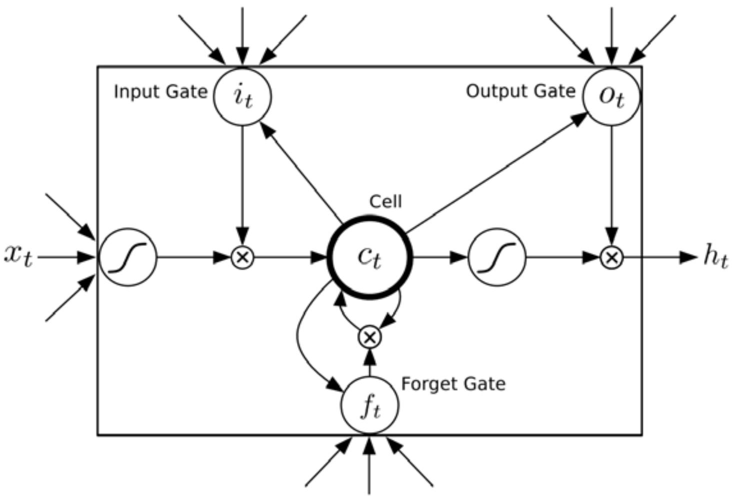
1. Numerical gradient check

```
fx = f(x) # evaluate function value at original point
grad = np.zeros_like(x)
# iterate over all indexes in x
it = np.nditer(x, flags=['multi_index'], op_flags=['readwrite'])
while not it.finished:

  # evaluate function at x+h
  ix = it.multi_index
  oldval = x[ix]
  x[ix] = oldval + h # increment by h
  fxph = f(x) # evalute f(x + h)
  x[ix] = oldval - h
  fxmh = f(x) # evaluate f(x - h)
  x[ix] = oldval # restore

  # compute the partial derivative with centered formula
  grad[ix] = (fxph - fxmh) / (2 * h) # the slope
  if verbose:
    print ix, grad[ix]
  it.iternext() # step to next dimension
```

Tricks

1. Numerical gradient check
2. Modulize layers: only three functions needed
   (1) output=forward(input,model)
   (2) dJ_dW=computeParamGrad(input,outputGrad,model)
   (3) dJ_dInput=computeInputGrad(input,outputGrad,model)
   Everything else is just putting together lego pieces





Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial

Questions?

Thanks!

Most materials taken from Andrej Karpathy/Richard Socher/Nando de Freitas/caffe CVPR tutorial