Introduction to Neural Networks

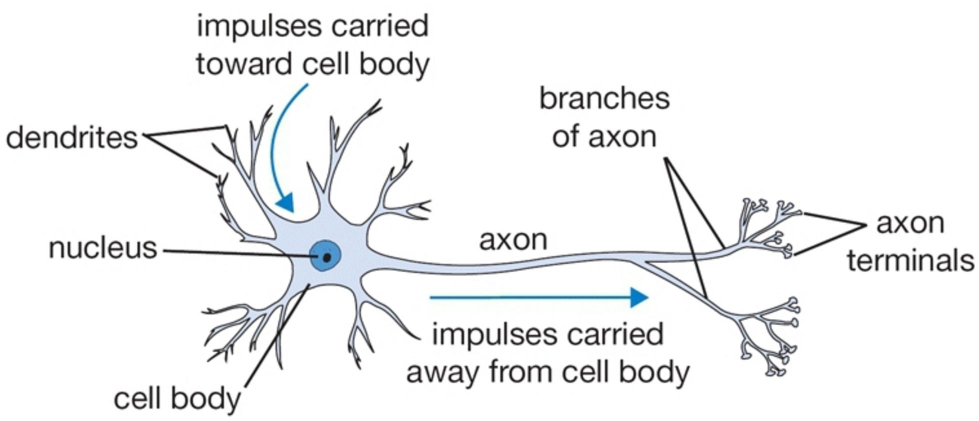
Machine Learning and Object Recognition 2016-2017

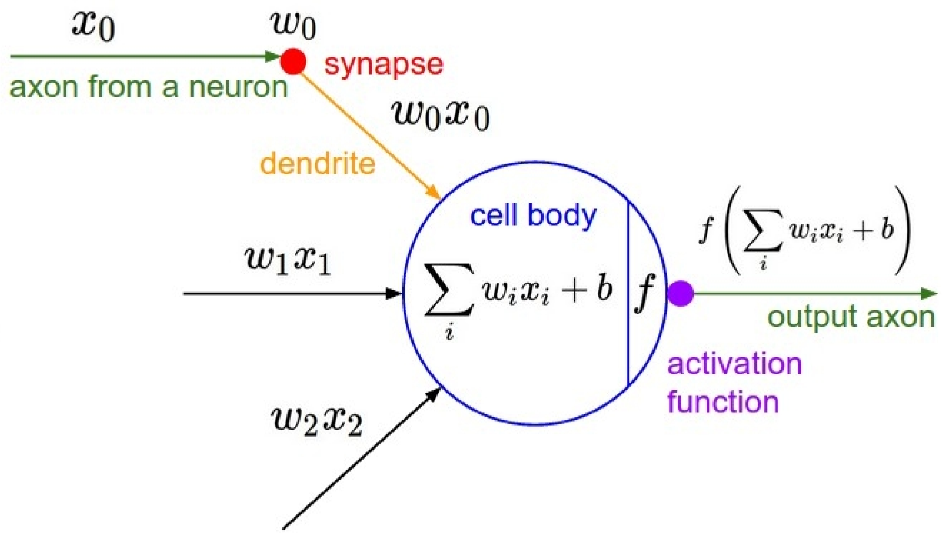
Course website:

http://thoth.inrialpes.fr/~verbeek/MLOR.16.17.php

Biological motivation

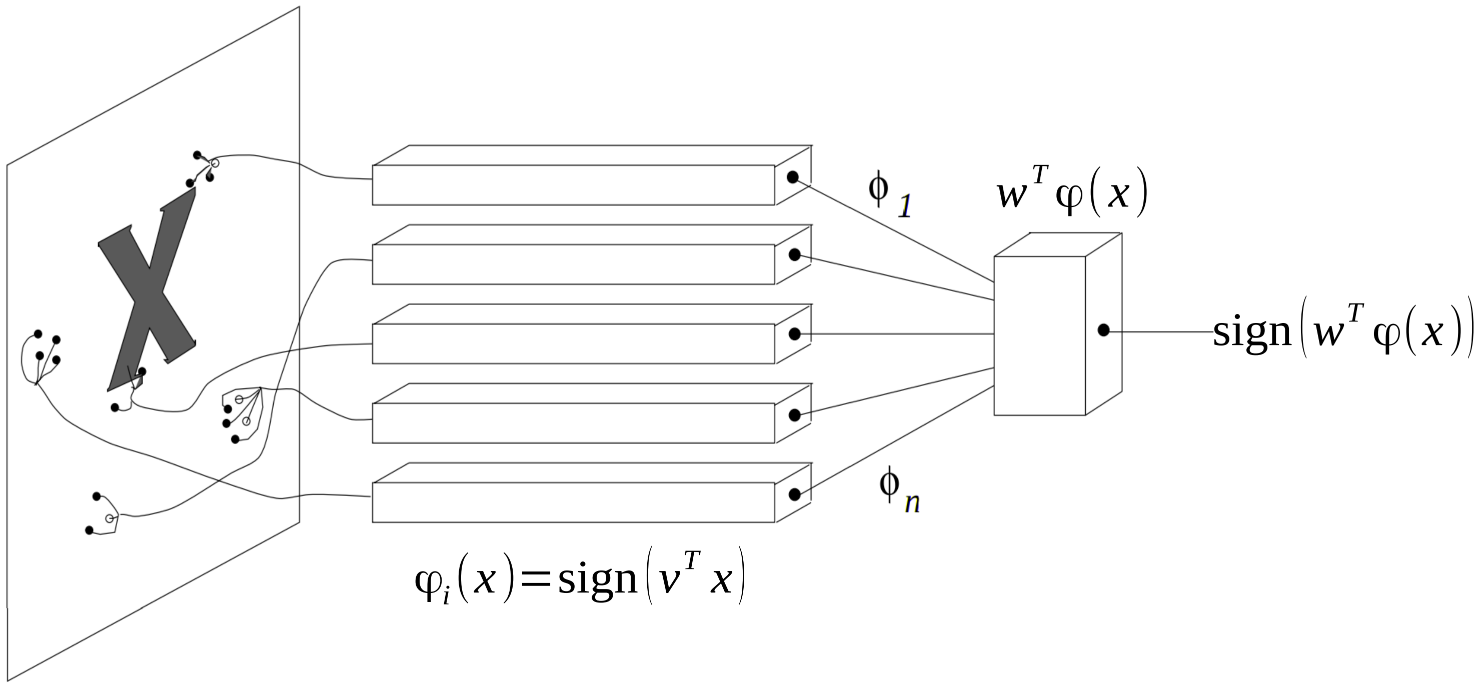
- Neuron is basic computational unit of the brain
  - about 10^11 neurons in human brain
- Simplified neuron model as linear threshold unit (McCulloch & Pitts, 1943)
  - Firing rate of electrical spikes modeled as continuous output quantity
  - Multiplicative interaction of input and connection strength (weight)
  - Multiple inputs accumulated in cell activation
  - Output is non linear function of activation
- Basic component in neural circuits for complex tasks





Rosenblatt's Perceptron

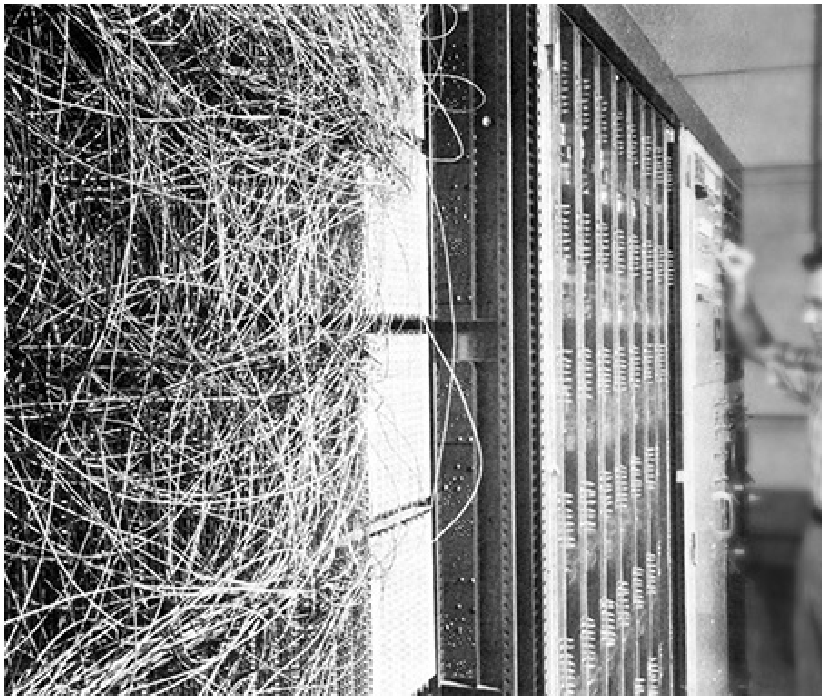
- One of the earliest works on artificial neural networks: 1957
  - Computational model of natural neural learning



- Binary classification based on sign of generalized linear function
  - Weight vector w learned using special purpose machines
  - Associative units in firs layer fixed by lack of learning rule at the time

Rosenblatt's Perceptron

20x20 pixel sensor



Random wiring of associative units

Rosenblatt's Perceptron

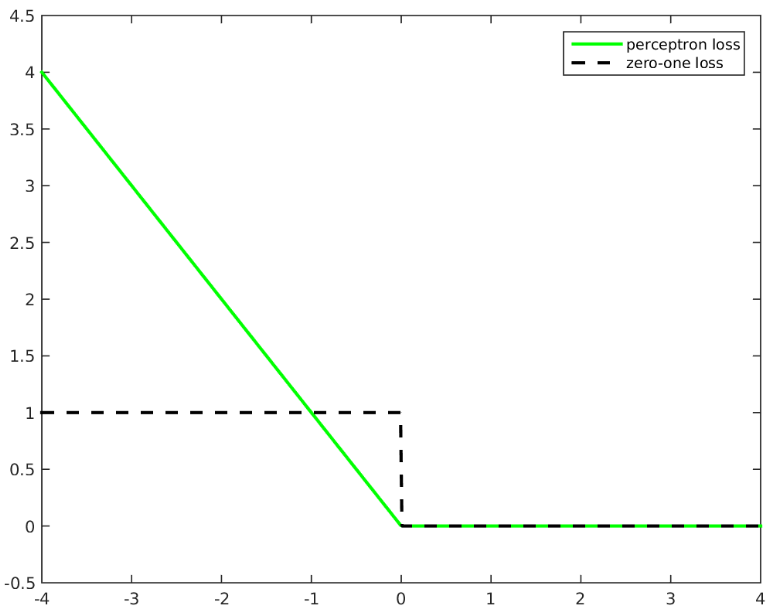
- Objective function linear in score over misclassified patterns $\quad t_i \in \{-1, +1\}$

$$E(w) = -\sum_{t_i \neq sign(f(x_i))} t_i f(x_i) = \sum_i max(0, -t_i f(x_i))$$

- Perceptron learning via stochastic gradient descent

$$w^{n+1} = w^n + \eta \times t_i \varphi(x_i) \times [t_i f(x_i) < 0]$$

  - Eta is the learning rate



Potentiometers as weights, adjusted by motors during learning

Limitations of the Perceptron

- Perceptron convergence theorem (Rosenblatt, 1962) states that
  - If training data is linearly separable, then learning algorithm will find a solution in a finite number of iterations
  - Faster convergence for larger margin (at fixed data scale)

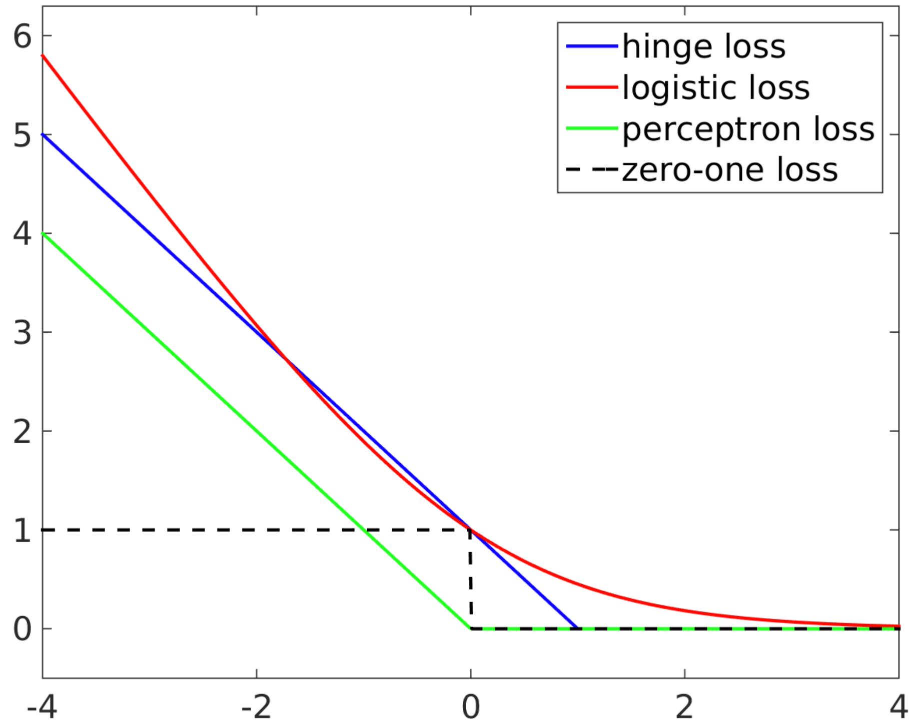
- If training data is linearly separable then the found solution will depend on the initialization and ordering of data in the updates
- If training data is not linearly separable, then the perceptron learning algorithm will not converge

- No direct multi-class extension
- No probabilistic output or confidence on classification

Relation to SVM and logistic regression

- Perceptron loss similar to hinge loss without the notion of margin
  - Cost function is not a bound on the zero-one loss
- All are either based on linear function or generalized linear function by relying on pre-defined non-linear data transformation

$$f(x) = w^T \varphi(x)$$



Kernels to go beyond linear classification

- Representer theorem states that in all these cases optimal weight vector is linear combination of training data

$$w = \sum_i \alpha_i \varphi(x_i)$$

$$f(x) = w^T \varphi(x) = \sum_i \alpha_i \langle \varphi(x_i), \varphi(x) \rangle$$

- Kernel trick allows us to compute dot-products between (high-dimensional) embedding of the data

$$k(x_i, x) = \langle \varphi(x_i), \varphi(x) \rangle$$

- Classification function is linear in data representation given by kernel evaluations over the training data

$$f(x) = \sum_i \alpha_i k(x, x_i) = \alpha^T k(x, .)$$

Limitation of kernels

- Classification based on weighted "similarity" to training samples
  - Design of kernel based on domain knowledge and experimentation

$$f(x) = \sum_i \alpha_i k(x, x_i) = \alpha^T k(x,.)$$

  - Some kernels are data adaptive, for example the Fisher kernel
  - Still kernel is designed before and separately from classifier training

- Number of free variables grows linearly in the size of the training data
  - Unless a finite dimensional explicit embedding is available $\varphi(x)$
  - Sometimes kernel PCA is used to obtain such a explicit embedding

- Alternatively: fix the number of "basis functions" in advance
  - Choose a family of non-linear basis functions
  - Learn the parameters, together with those of linear function

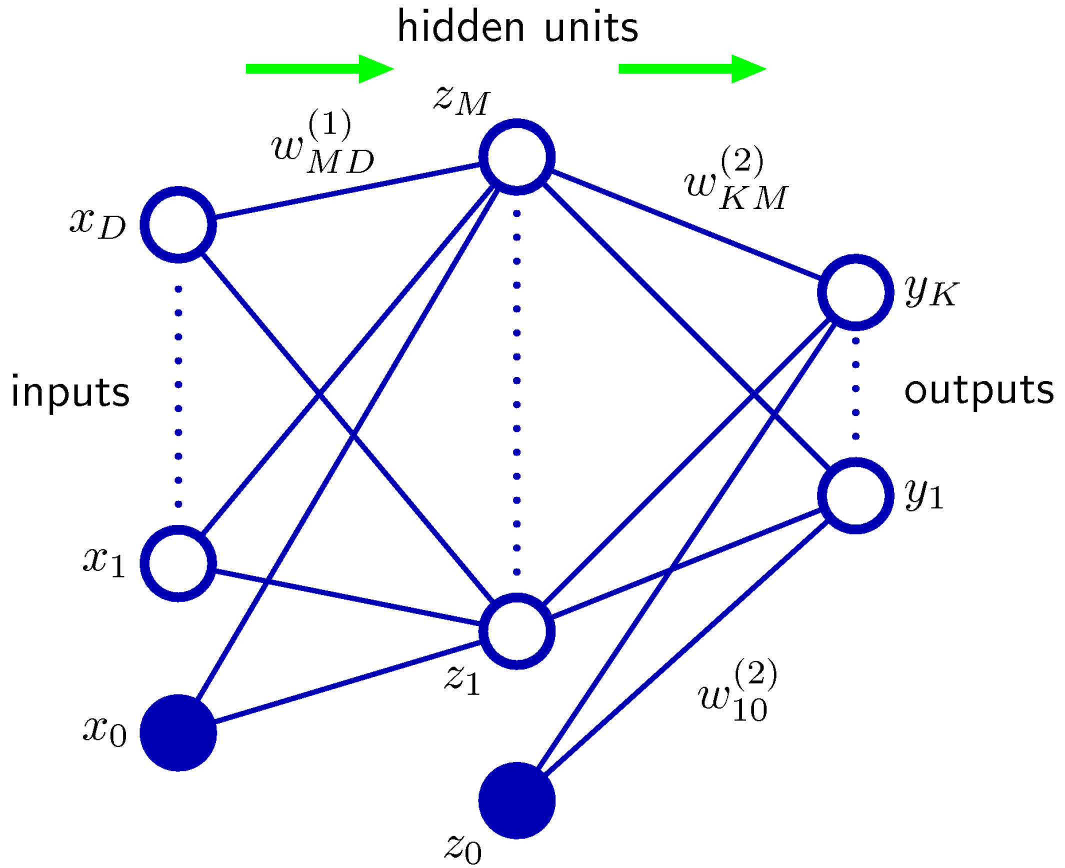
$$f(x) = \sum_i \alpha_i \varphi_i(x; \theta_i)$$

Feed-forward neural networks

- Define outputs of one layer as scalar non-linearity of linear function of input
- Known as “multi-layer perceptron”
  - Perceptron has a step non-linearity of linear function (historical)
  - Other non-linearities are used in practice (see below)

$$z_j = h\left(\sum_i x_i w_{ij}^{(1)}\right)$$

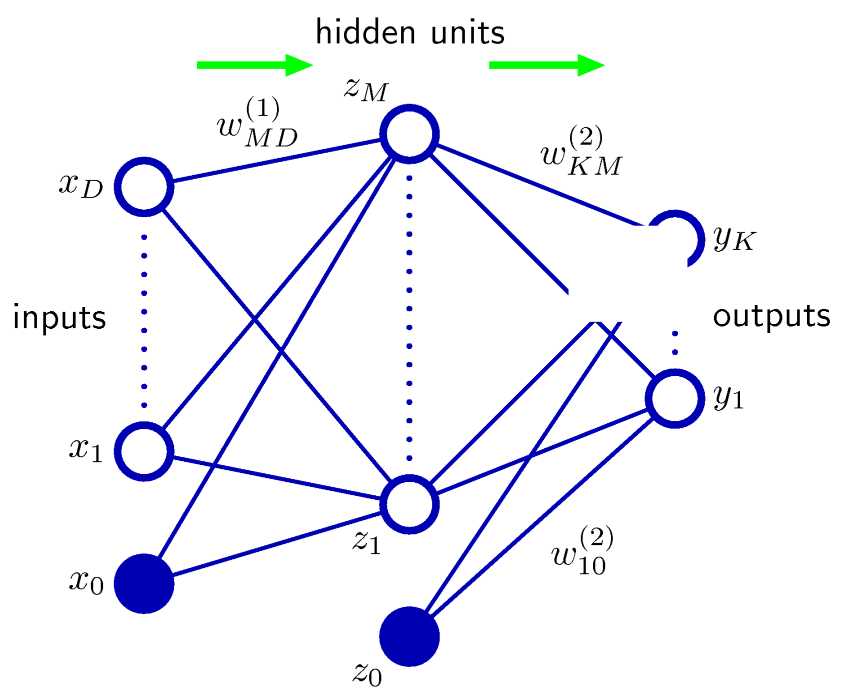
$$y_k = \sigma\left(\sum_j z_j w_{jk}^{(2)}\right)$$



Feed-forward neural networks

- If "hidden layer" activation function is taken to be linear than a single-layer linear model is obtained

- Two-layer networks can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy provided the network has a sufficiently large number of hidden units
  - Holds for many non-linearities, but not for polynomials



Classification over binary inputs

- Consider simple case with binary units
  - Inputs and activations are all +1 or -1
  - Total number of inputs is 2^D
  - Classification problem into two classes
- Use a hidden unit for each positive sample x_m

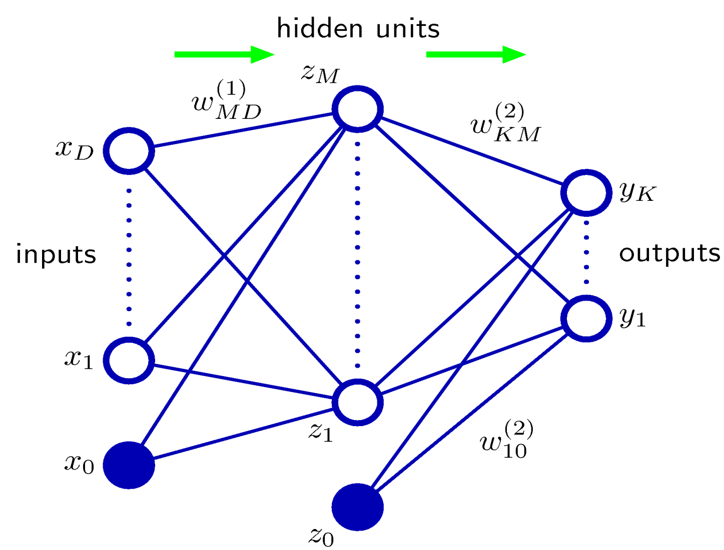
$$z_m = \text{sign}\left(\sum_{i=1}^{D} w_{mi} x_i - D + 1\right)$$

$$w_{mi} = x_{mi}$$

  - Activation is +1 if and only if input is x_m
- Let output implement an "or" over hidden units

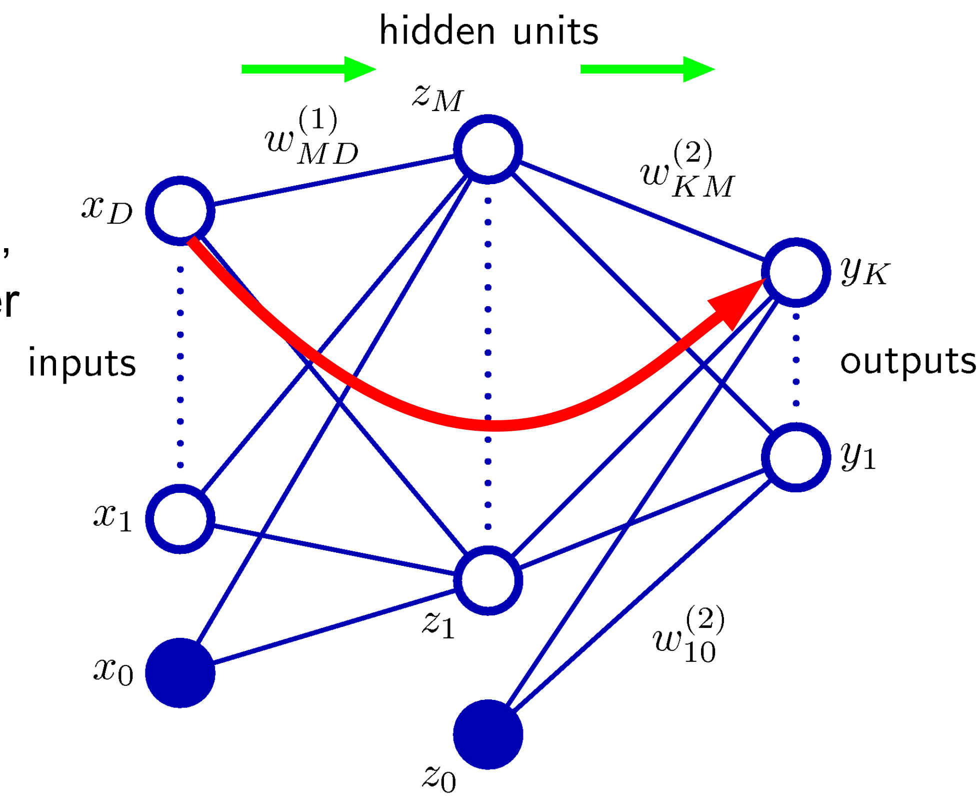
$$y = \text{sign}\left(\sum_{m=1}^{M} z_m + M - 1\right)$$

- Problem: may need exponential number of hidden units



Feed-forward neural networks

- Architecture can be generalized
  - More than two layers of computation
  - Skip-connections from previous layers
- Feed-forward nets are restricted to directed acyclic graphs of connections
  - Ensures that output can be computed from the input in a single feed-forward pass from the input to the output
- Main issues:
  - Designing network architecture
    - Nr nodes, layers, non-linearities, etc
  - Learning the network parameter
    - Non-convex optimization



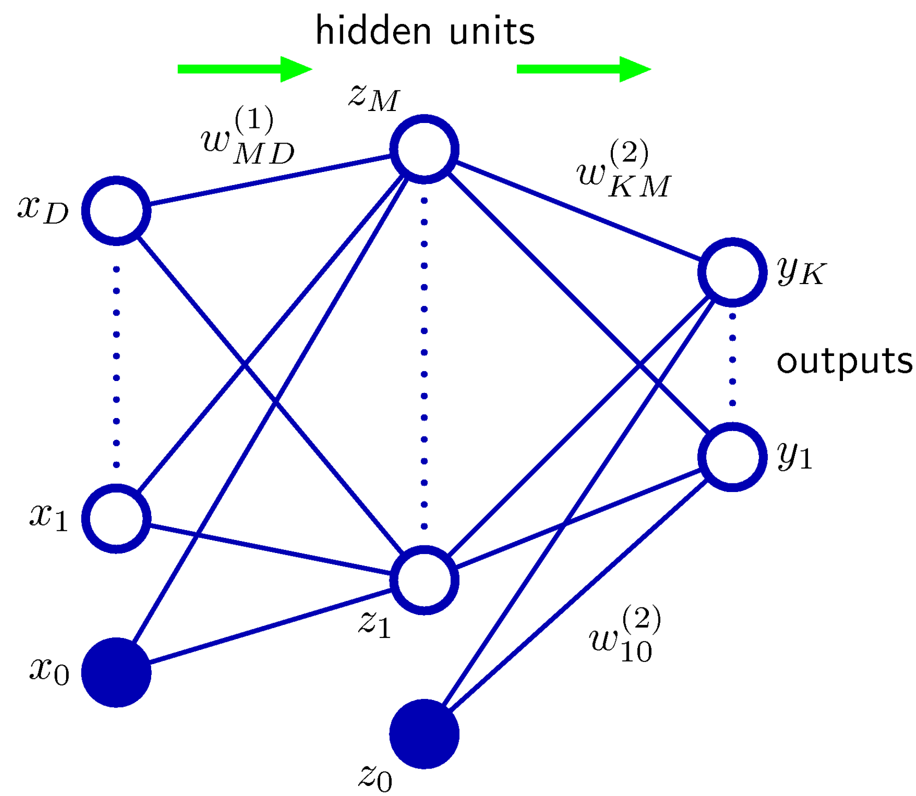
An example: multi-class classifiction

- One output score for each target class
- Multi-class logistic regression loss
  - Define probability of classes by softmax over scores
  - Maximize log-probability of correct class

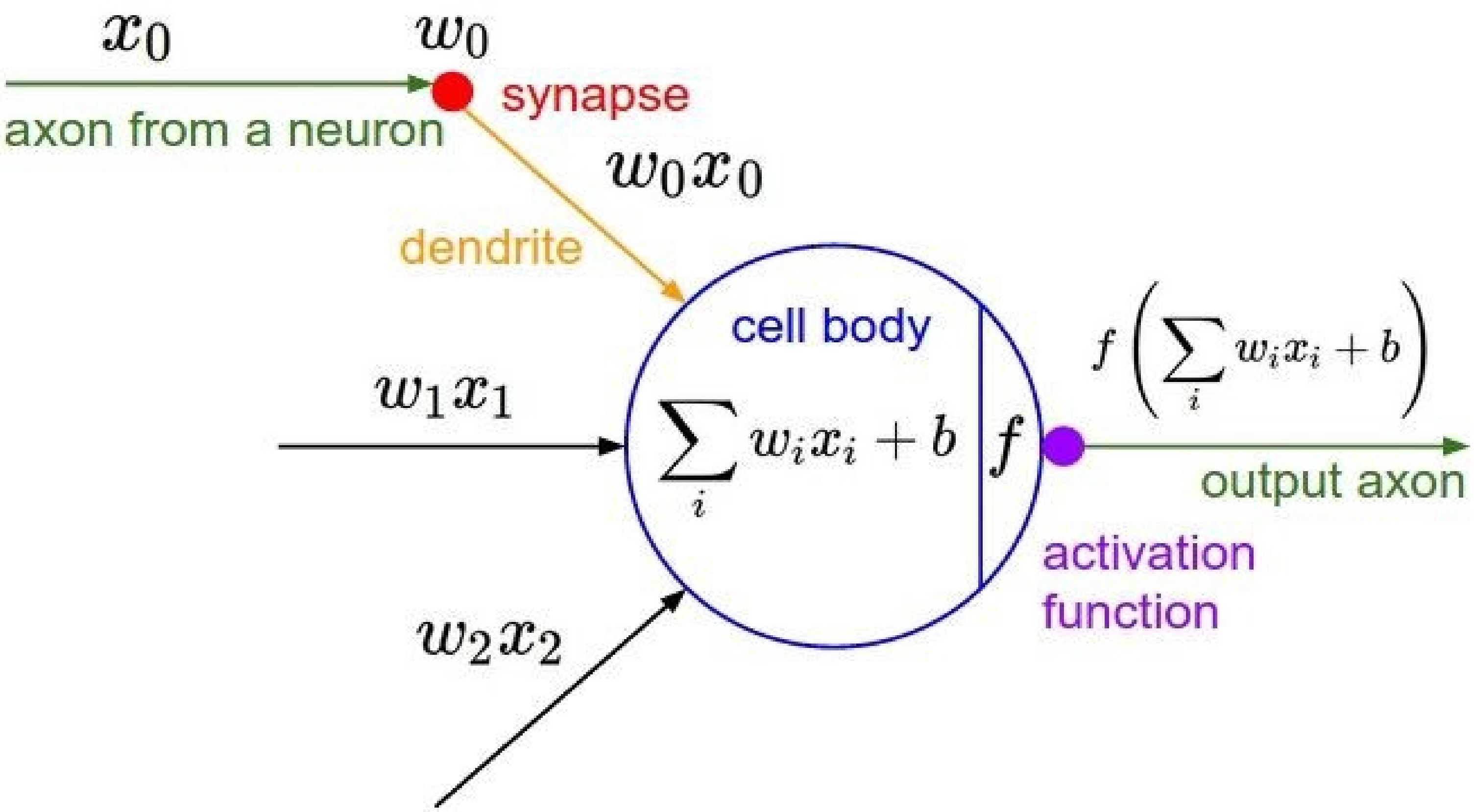
$$p(y=c|x)=\frac{\exp y_c}{\sum_k \exp y_k}$$

- Precisely as before, but we are now learning the data representation concurrently with the linear classifier

- Representation learning in discriminative and coherent manner
- Fisher kernel also data adaptive but not discriminative and task dependent
- More generally, we can choose a loss function for the problem of interest and optimize all network parameters w.r.t. this objective (regression, metric learning, ...)

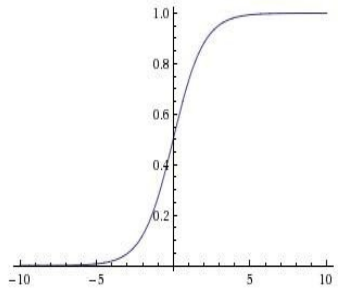


Activation functions

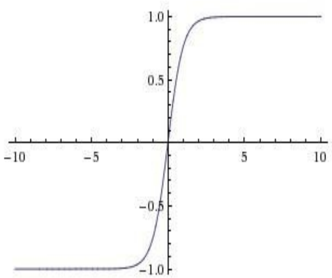


Activation functions

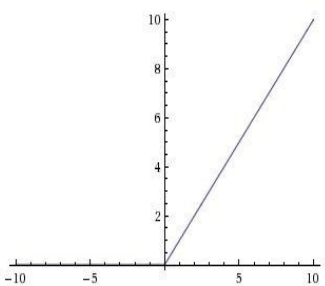
Sigmoid $1/(1+e^{-x})$



tanh

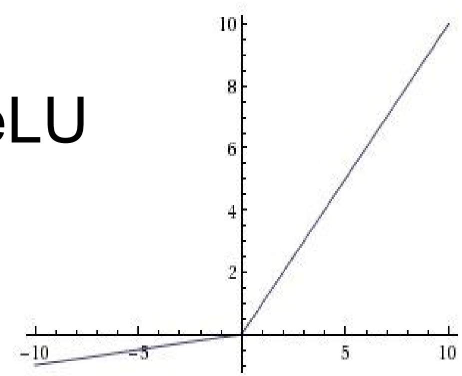


ReLU $\max(0, x)$

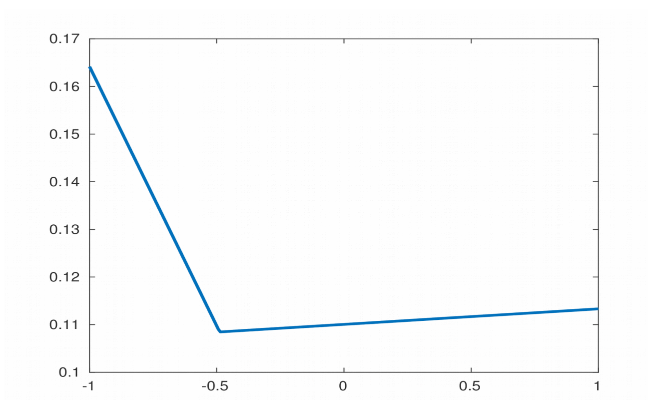


Leaky ReLU

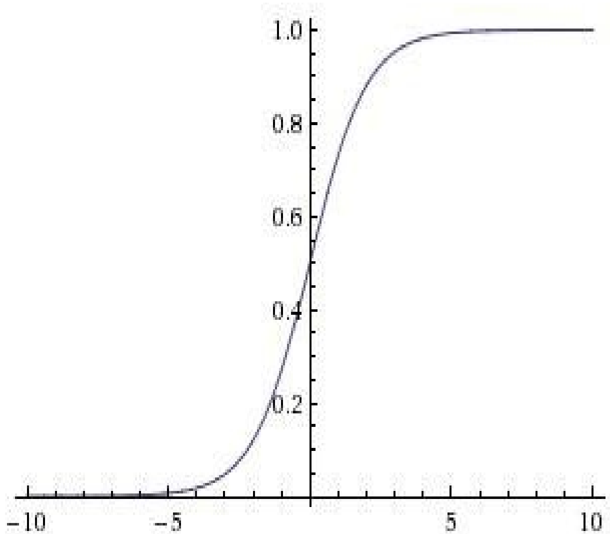
$\max(\alpha x, x)$



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



Activation Functions



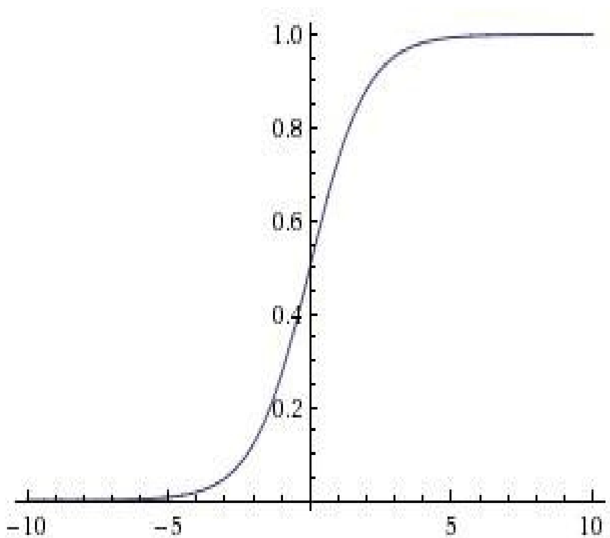
Sigmoid

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

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



Sigmoid

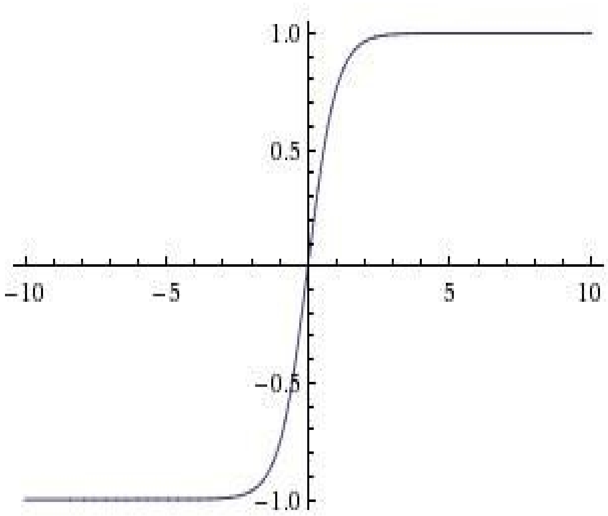
$$\sigma(x) = 1/(1+e^{-x})$$

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



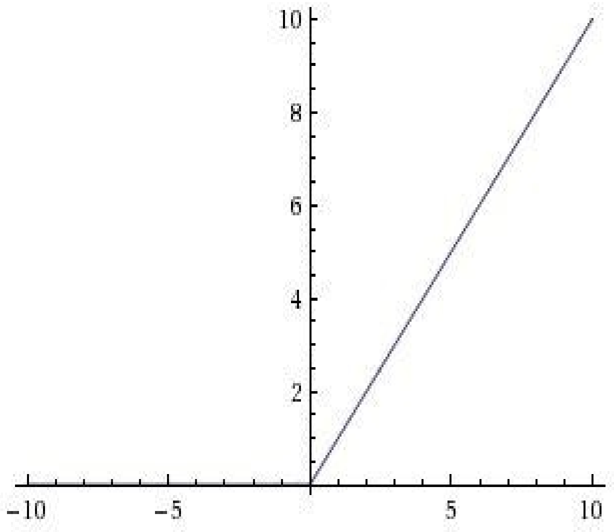
tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

[LeCun et al., 1991]

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



ReLU
(Rectified Linear Unit)

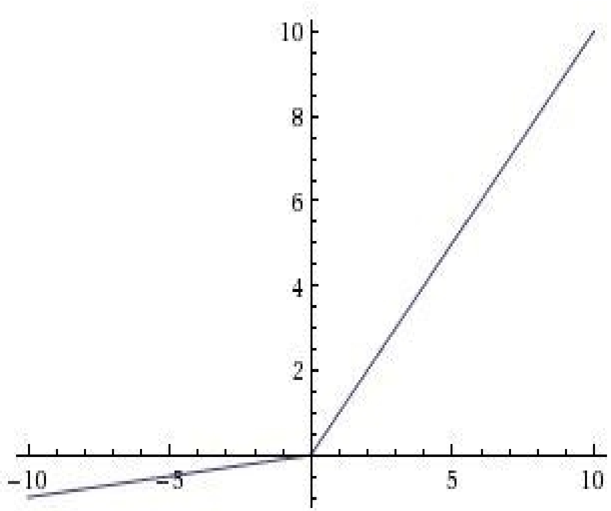
Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

[Nair & Hinton, 2010]

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



Leaky ReLU

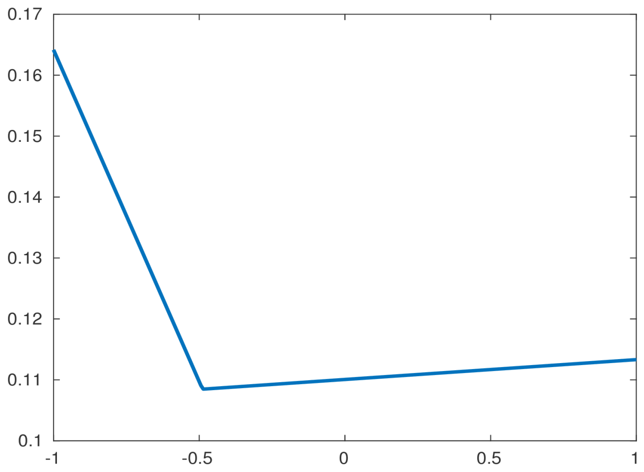
- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- will not “die”.

$$f(x) = \max(0.01x, x)$$

[Mass et al., 2013] [He et al., 2015]

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



- Does not saturate
- Computationally efficient
- Will not "die"
- Maxout networks can implement ReLU networks and vice-versa
- More parameters per node

Maxout

$$\max\left(w_1^T x, w_2^T x\right)$$

[Goodfellow et al., 2013]

Training feed-forward neural network

- Non-convex optimization problem in general (or at least in useful cases)
  - Typically number of weights is (very) large (millions in vision applications)
  - Seems that many different local minima exist with similar quality

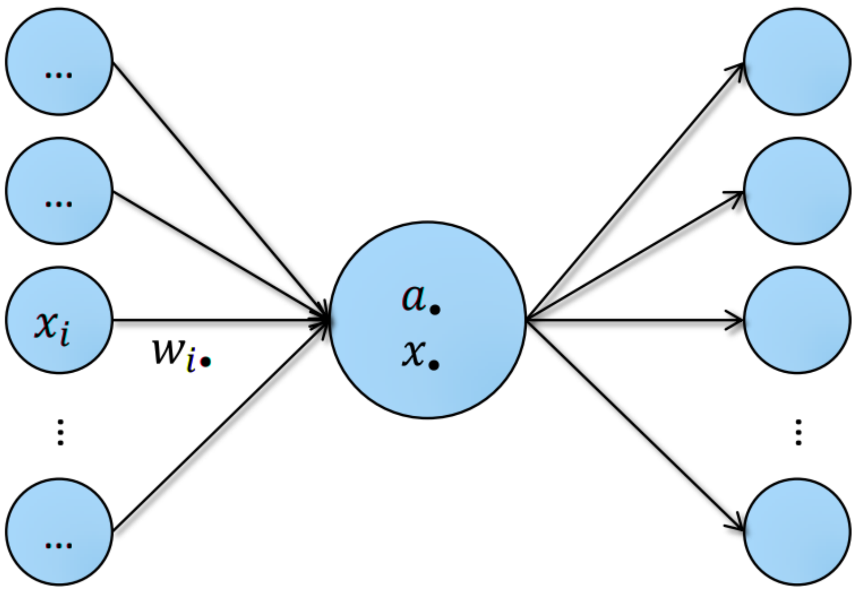
$$\frac{1}{N}\sum_{i=1}^{N} L\left(f\left(x_i\right), y_i; W\right) + \lambda\,\Omega\left(W\right)$$

- Regularization
  - L2 regularization: sum of squares of weights
  - “Drop-out”: deactivate random subset of weights in each iteration
    - Similar to using many networks with less weights (shared among them)

- Training using simple gradient descend techniques
  - Stochastic gradient descend for large datasets (large N)
  - Estimate gradient of loss terms by averaging over a relatively small number of samples

Training the network: forward propagation

- Forward propagation from input nodes to output nodes
  - Accumulate inputs into weighted sum
  - Apply scalar non-linear activation function f
- Use Pre(j) to denote all nodes feeding into j



$$a_j = \sum_{i \in Pre(j)} w_{ij} x_i$$

$$x_j = f(a_j)$$

Training the network: backward propagation

- Input aggregation and activation

$$a_j = \sum_{i \in Pre(j)} w_{ij} x_i$$

$$x_j = f(a_j)$$

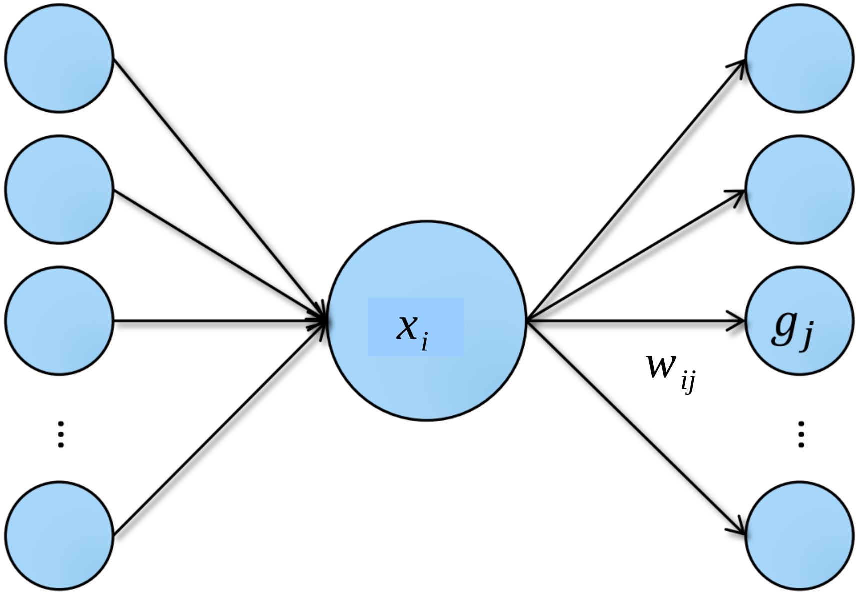
- Partial derivative of loss w.r.t. input

$$g_j = \frac{\partial L}{\partial a_j}$$

- Partial derivative w.r.t. learnable weights

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} = g_j x_i$$

- Gradient of weights between two layers given by outer-product of x and g



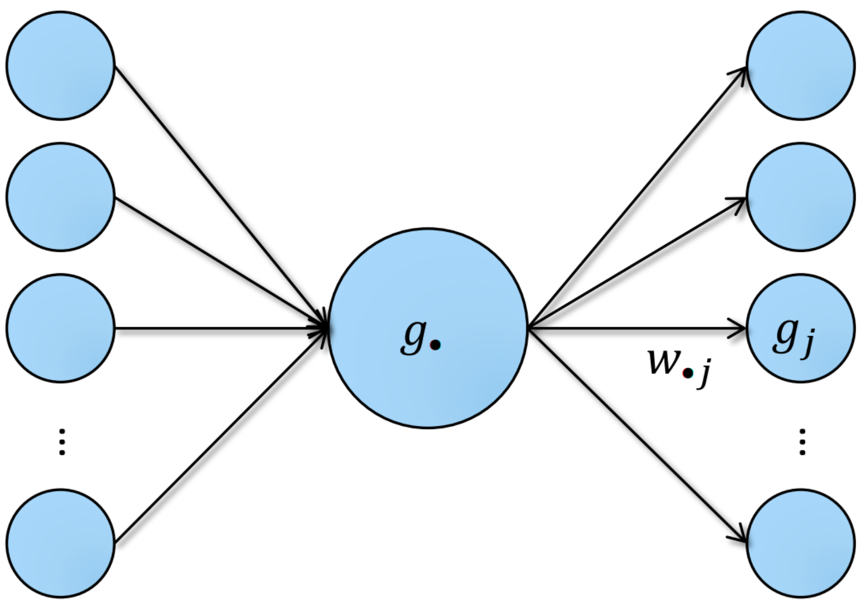
Training the network: backward propagation

- Backward propagation of loss gradient from output nodes to input nodes
  - Application of chainrule of derivatives
- Accumulate gradients from downstream nodes
  - Post(i) denotes all nodes that i feeds into
  - Weights propagate gradient back
- Multiply with derivative of local activation

$$a_j = \sum_{i \in Pre(j)} w_{ij} x_i$$

$$x_j = f(a_j)$$

$$g_i = \frac{\partial L}{\partial a_i}$$



$$\frac{\partial L}{\partial x_i} = \sum_{j \in Post(i)} \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial x_i}$$
$$= \sum_{j \in Post(i)} g_j w_{ij}$$

$$g_i = \frac{\partial x_i}{\partial a_i} \frac{\partial L}{\partial x_i}$$
$$= f'(a_i) \sum_{j \in Post(i)} w_{ij} g_j$$

Training the network: forward and backward propagation

- Special case for Rectified Linear Unit (ReLU) activations

$$f(a) = max(0, a)$$

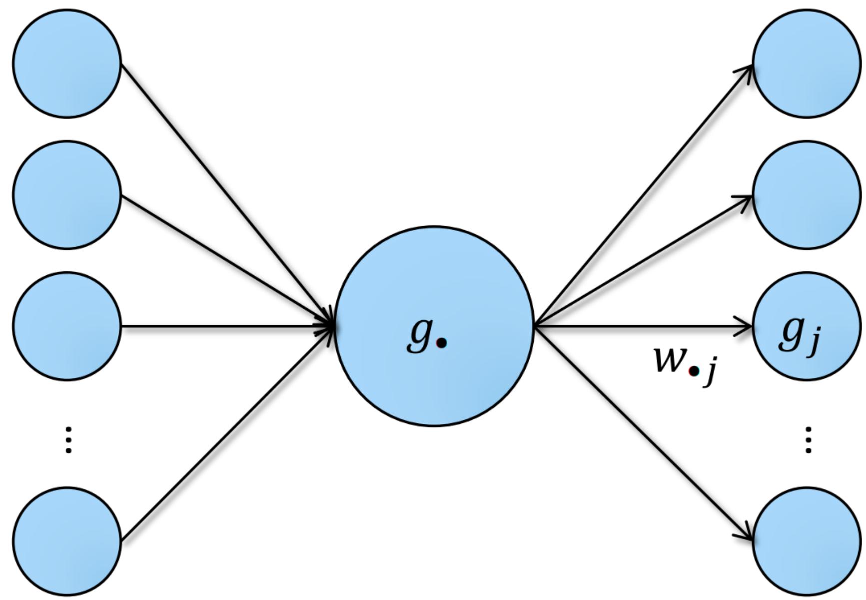
- Sub-gradient is step function

$$f'(a) = \begin{cases} 0 & \text{if } a \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

- Sum gradients from downstream nodes

$$g_i = \begin{cases} 0 & \text{if } a_i \leq 0 \\ \sum_{j \in Post(i)} w_{ij} g_j & \text{otherwise} \end{cases}$$

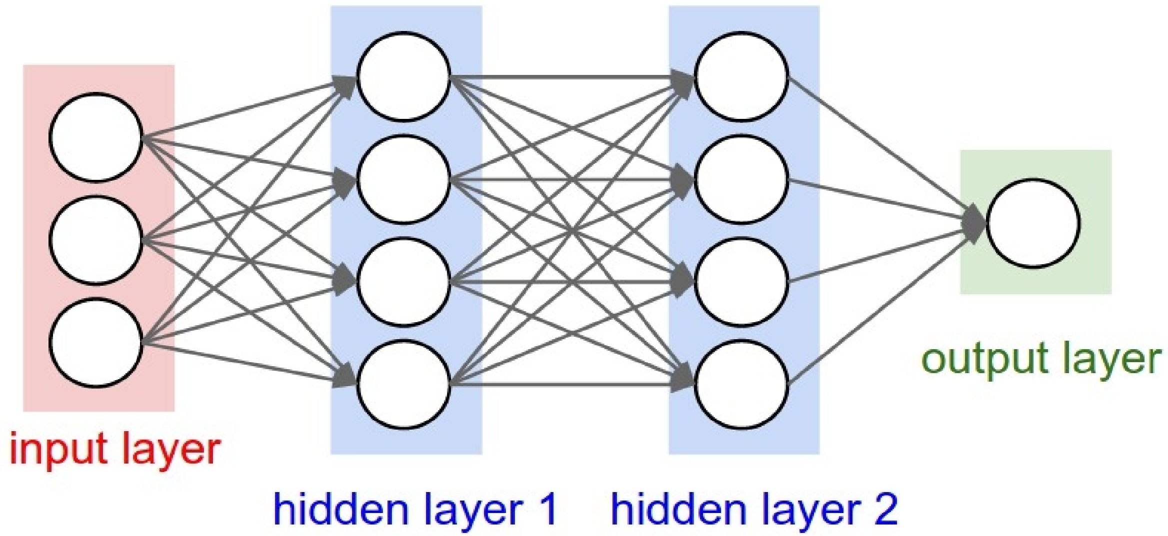
  - Set to zero if in ReLU zero-regime
  - Compute sum only for active units



- Note how gradient on incoming weights is "killed" by inactive units
  - Generates tendency for those units to remain inactive

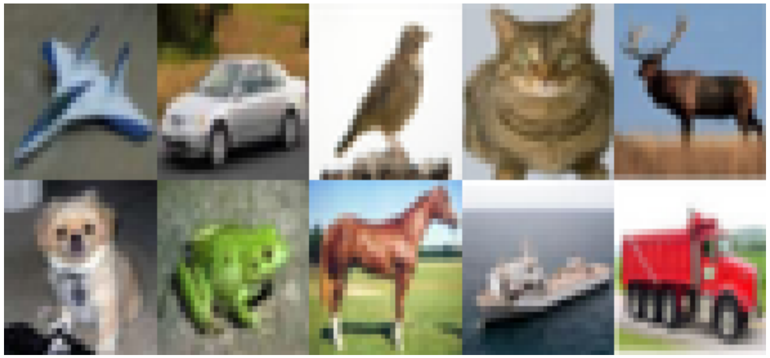
$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} = g_j x_i$$

Neural Networks



How to represent the image at the network input?

Input example : an image

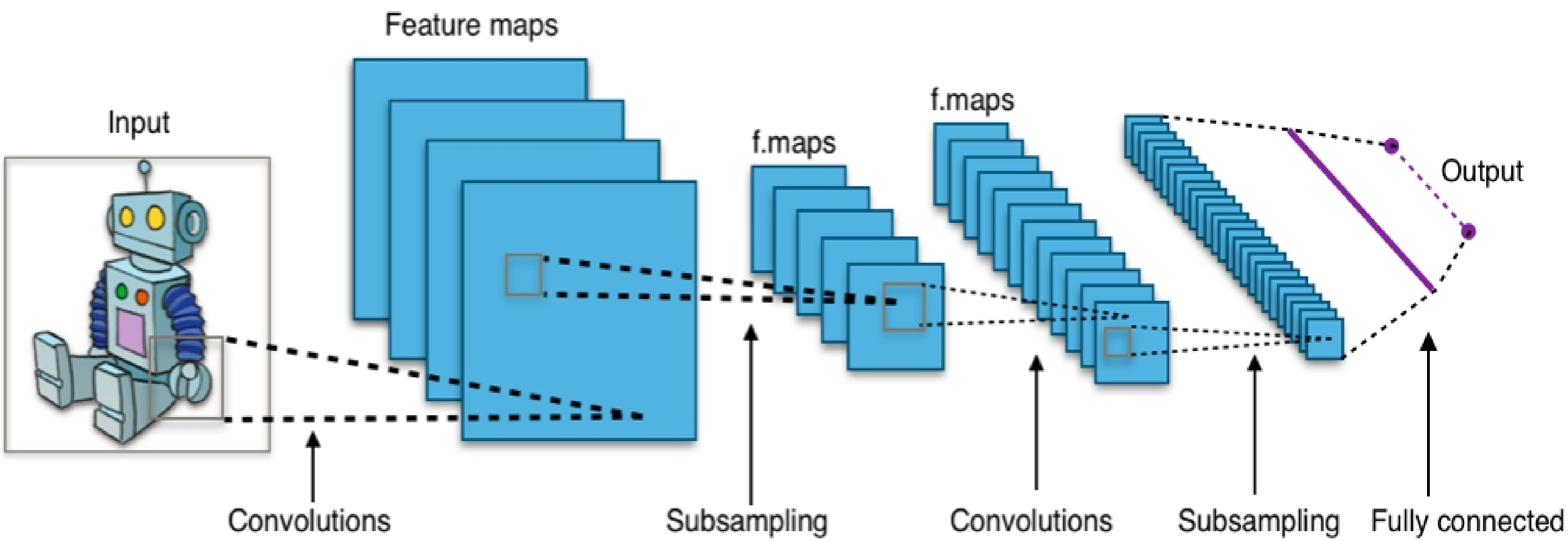


Output example : one class

airplane	dog
automobile	frog
bird	horse
cat	ship
deer	truck

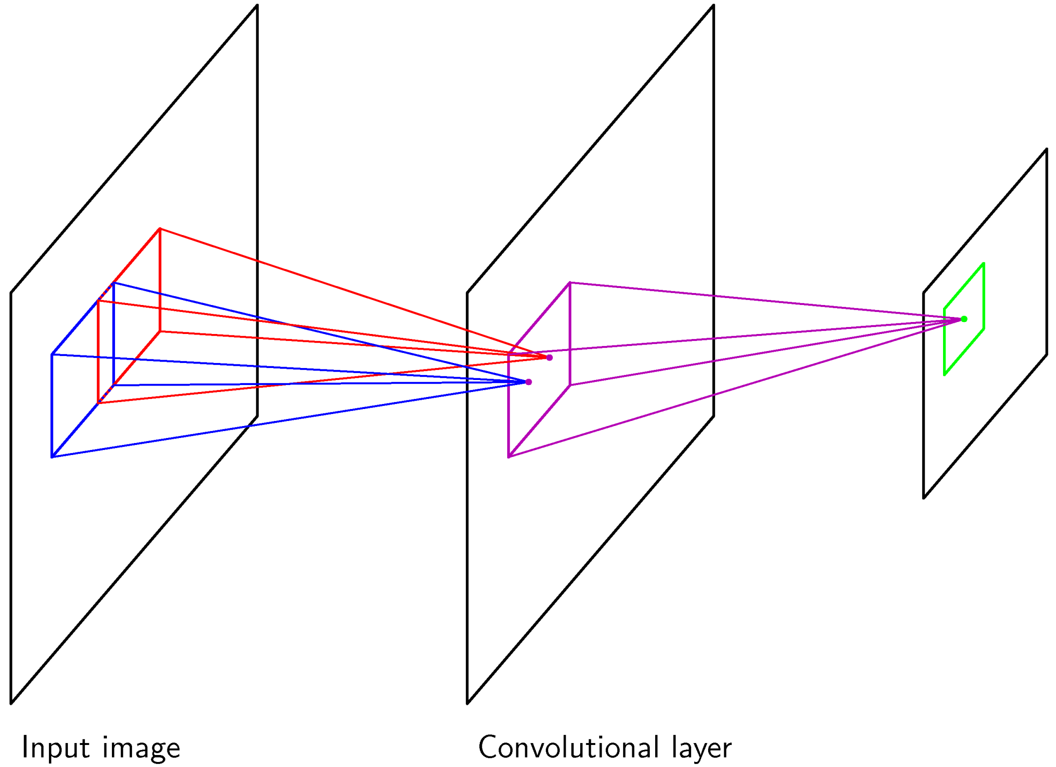
Convolutional neural networks

- A convolutional neural network is a feedforward network where
  - Hidden units are organizes into images or “response maps”
  - Linear mapping from layer to layer is replaced by convolution



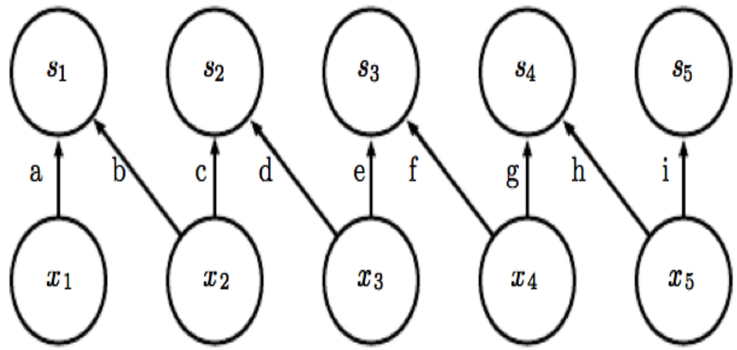
Convolutional neural networks

- Local connections: motivation from findings in early vision
  - Simple cells detect local features
  - Complex cells pool simple cells in retinotopic region
- Convolutions: motivated by translation invariance
  - Same processing should be useful in different image regions

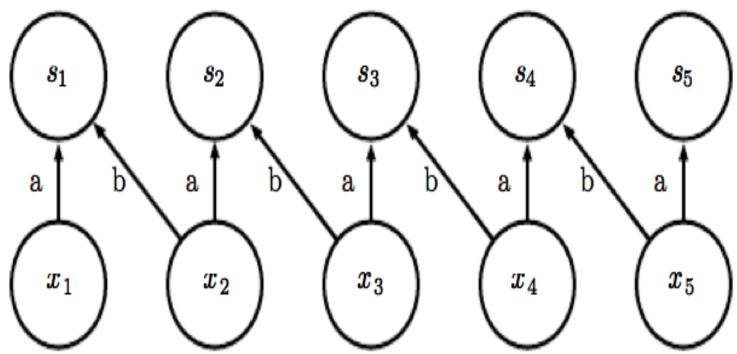


Local connectivity

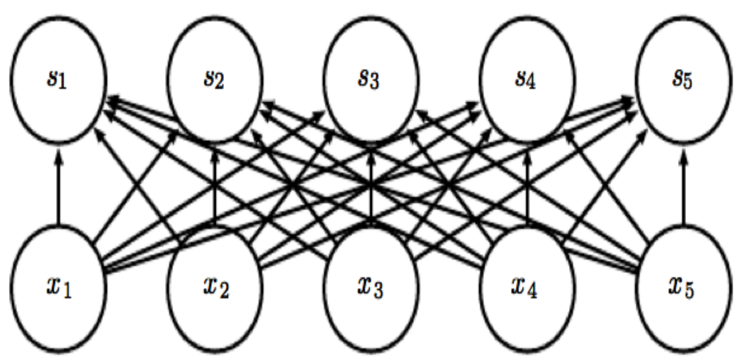
Locally connected layer



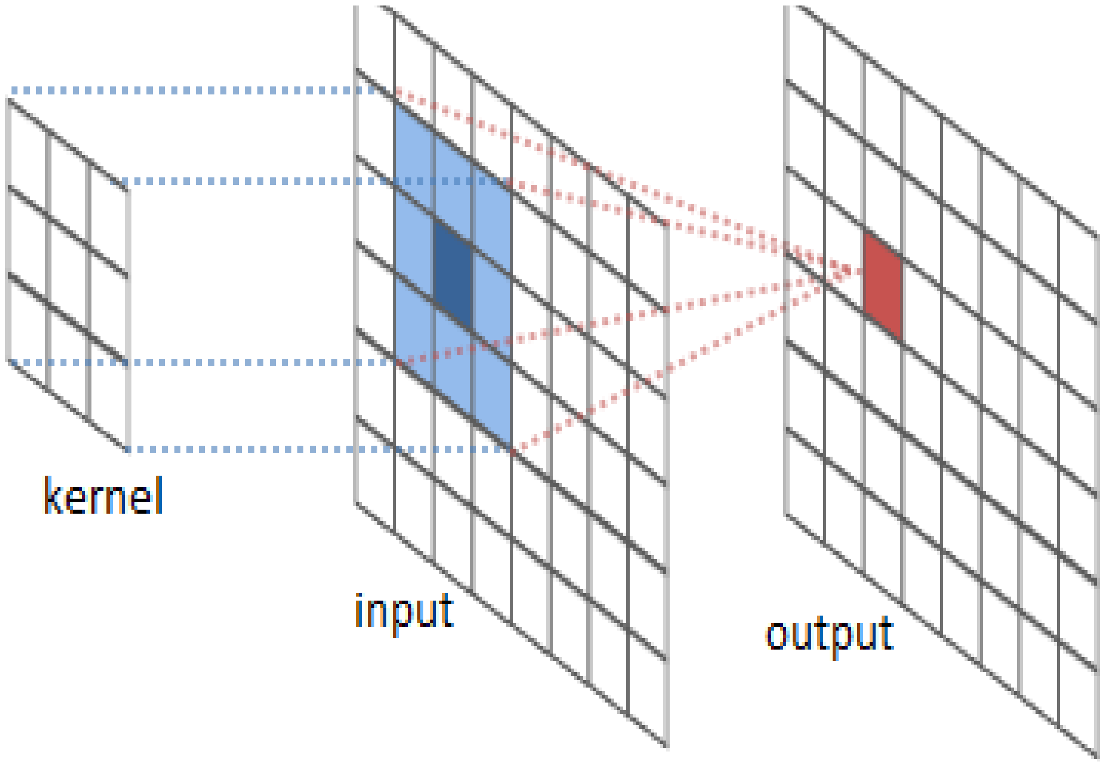
Convolutional layer



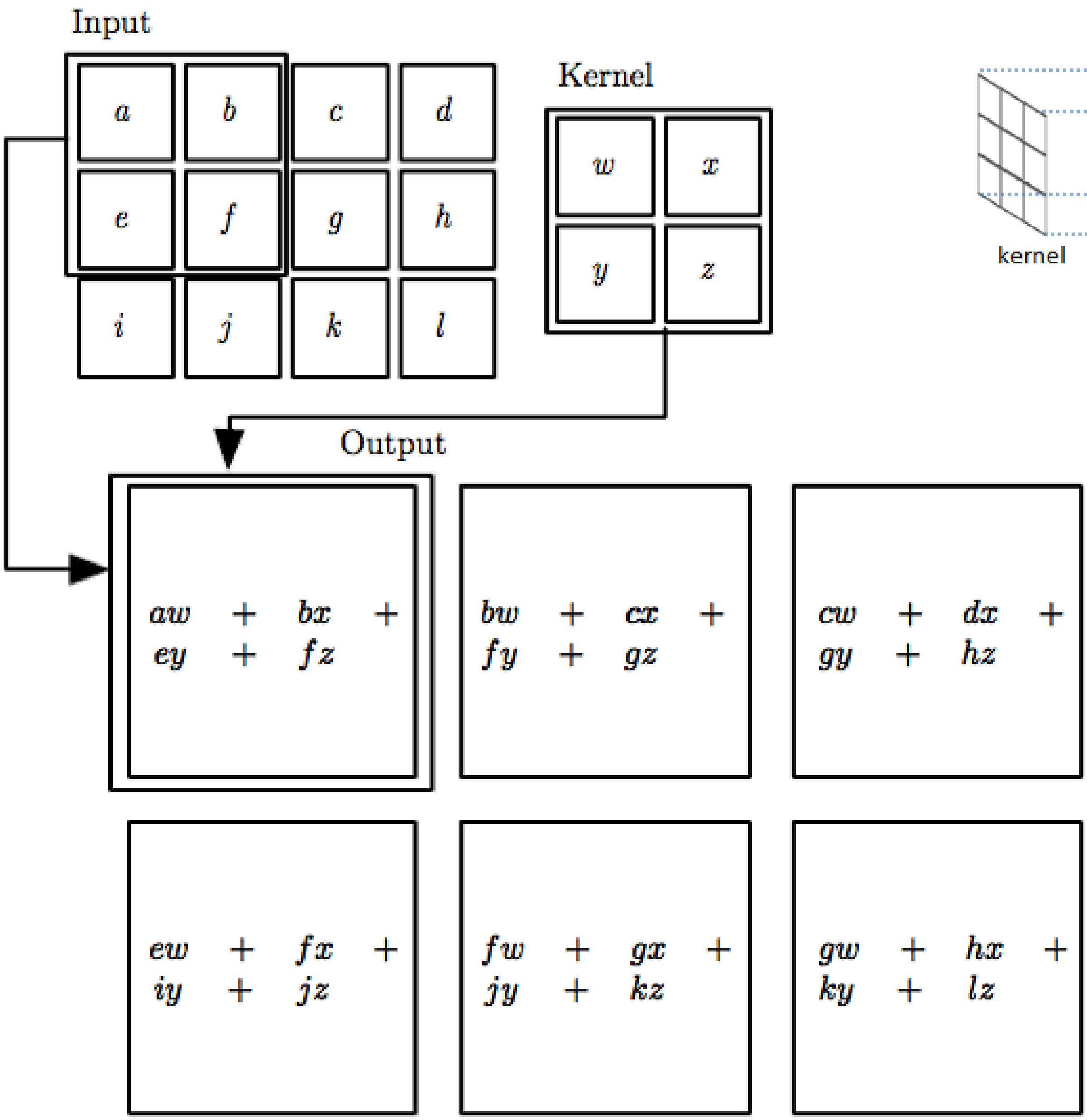
Fully connected layer

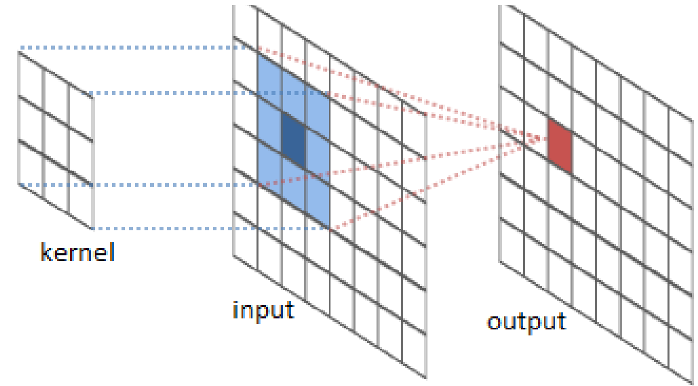


The convolution operation



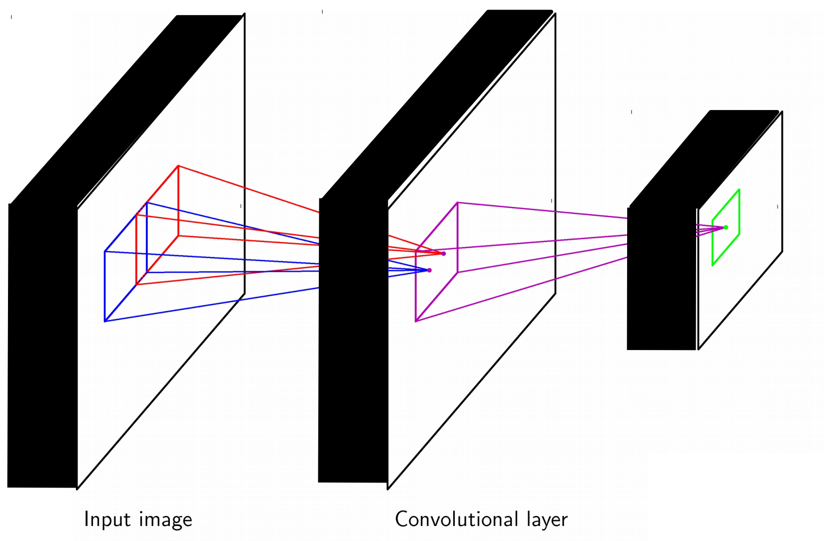
The convolution operation





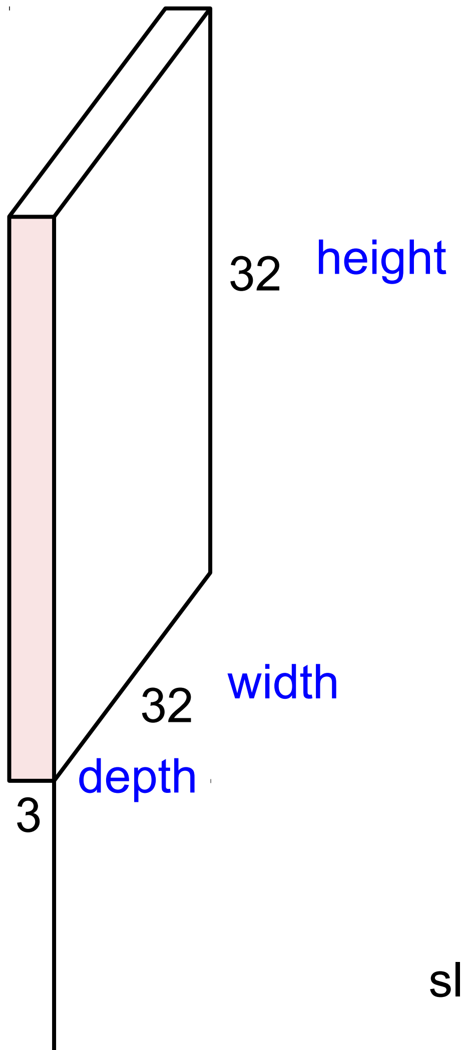
Convolutional neural networks

- Hidden units form another “image” or “response map”
  - Result of convolution: translation invariant linear funcion of local inputs
  - Followed by non-linearity
- Different convolutions can be computed “in parallel”
  - Gives a “stack” of response maps
  - Similarly, convolutional filters “read” across different maps
  - Input may also be multi-channel, e.g. RGB image
- Sharing of weights across hidden units
  - Number of parameters decoupled from input and representation size



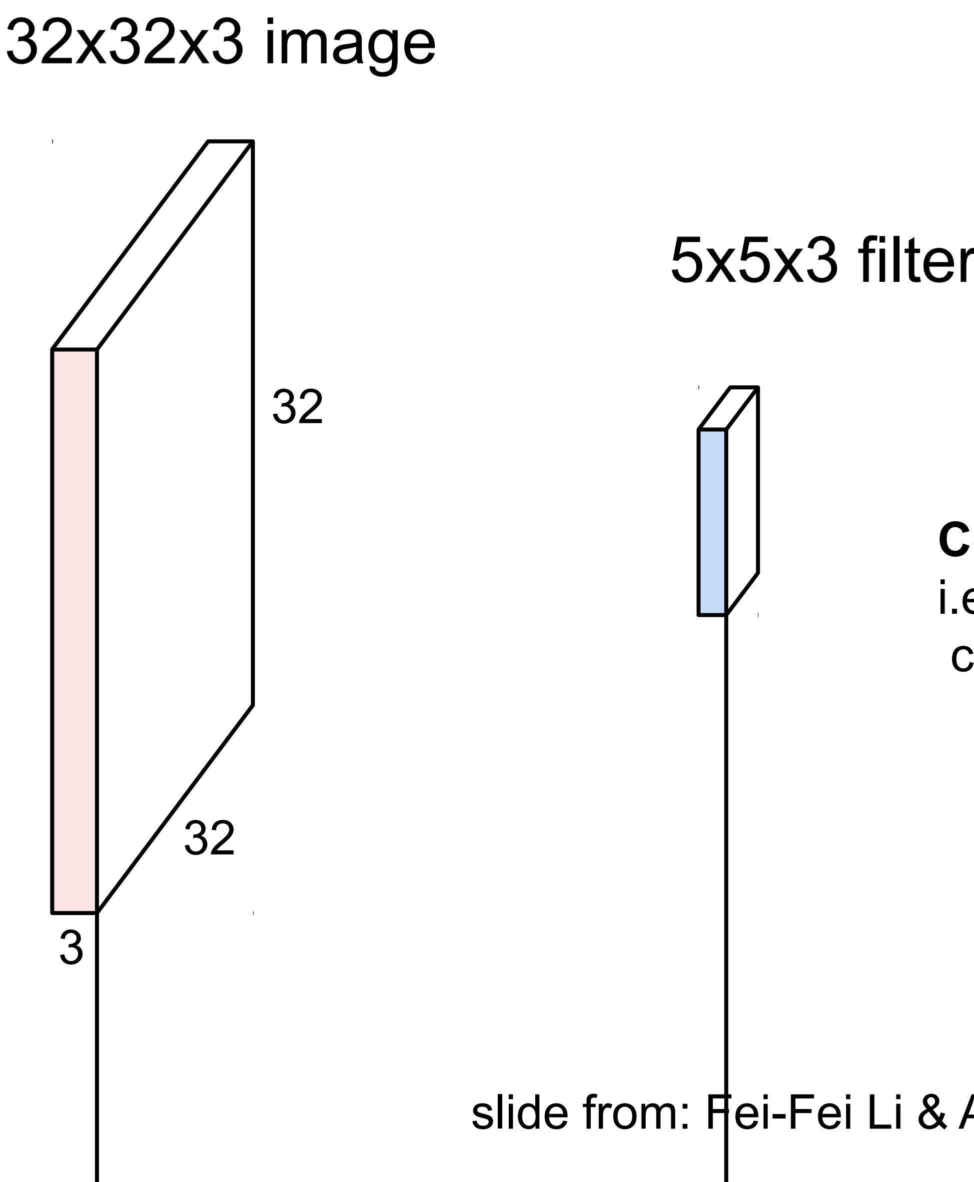
Convolution Layer

32x32x3 image



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

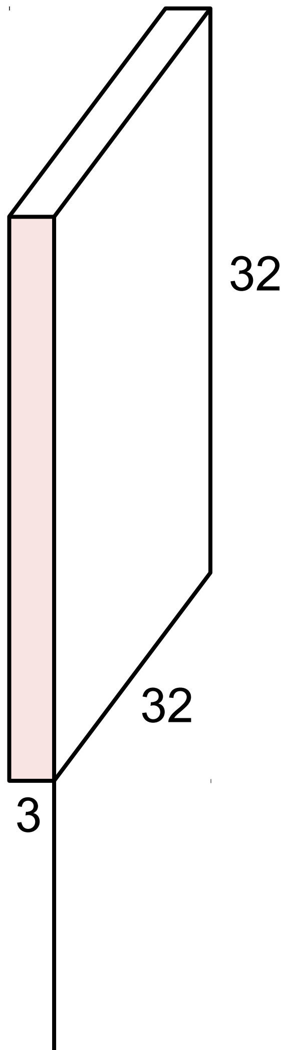


Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

32x32x3 image



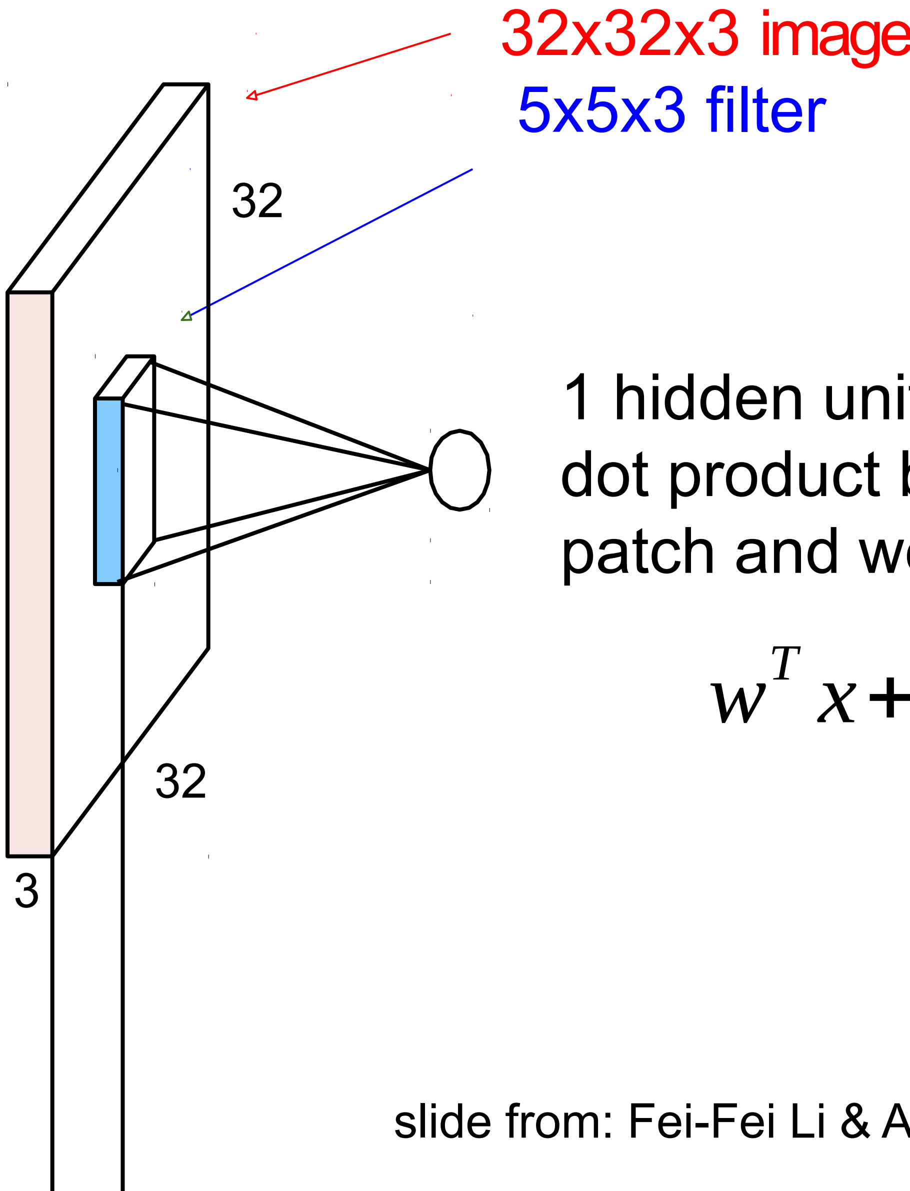
Filters always extend the full depth of the input volume

5x5x3 filter

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

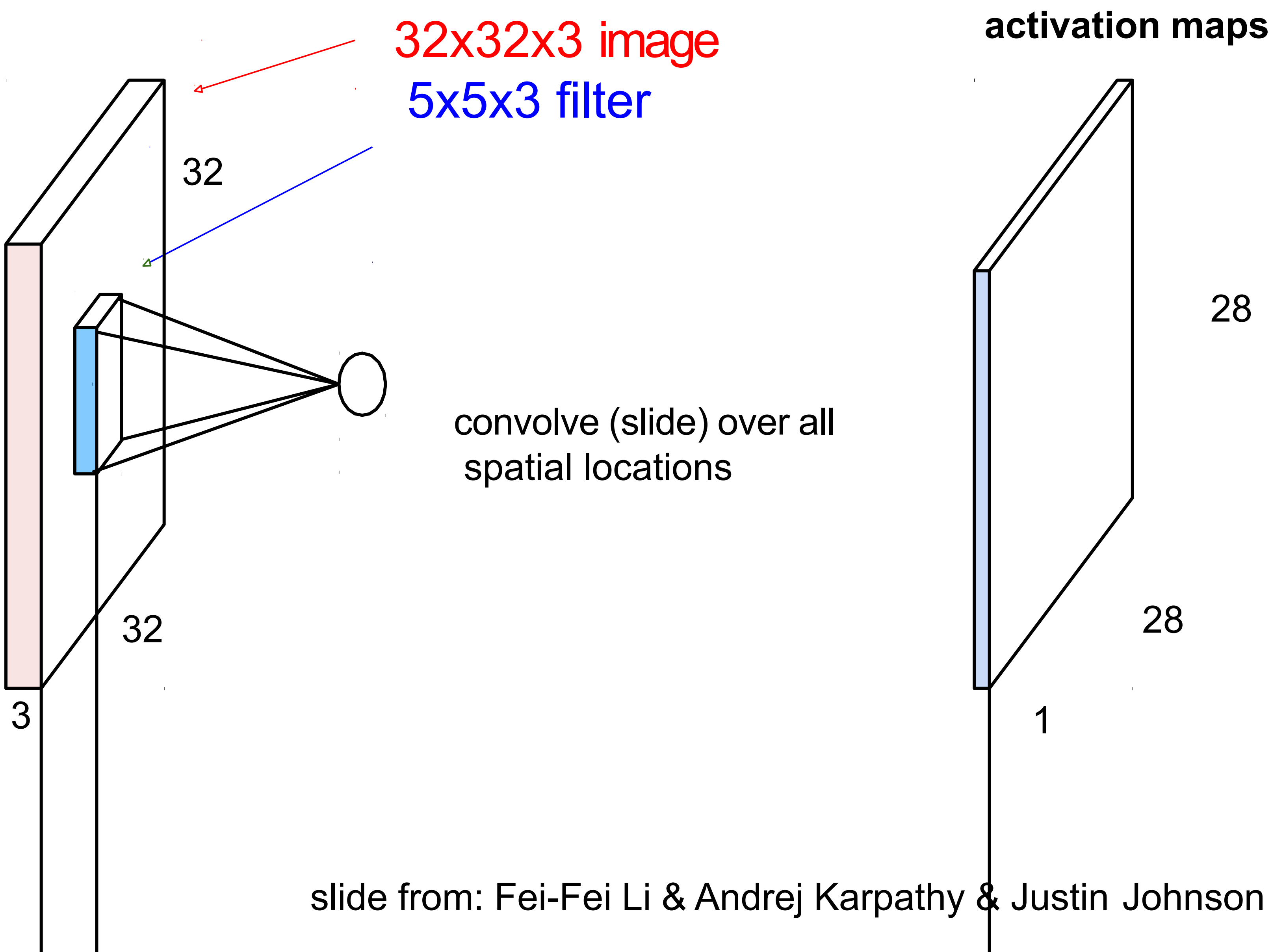


1 hidden unit:
dot product between 5x5x3=75 input patch and weight vector + bias

$$w^T x + b$$

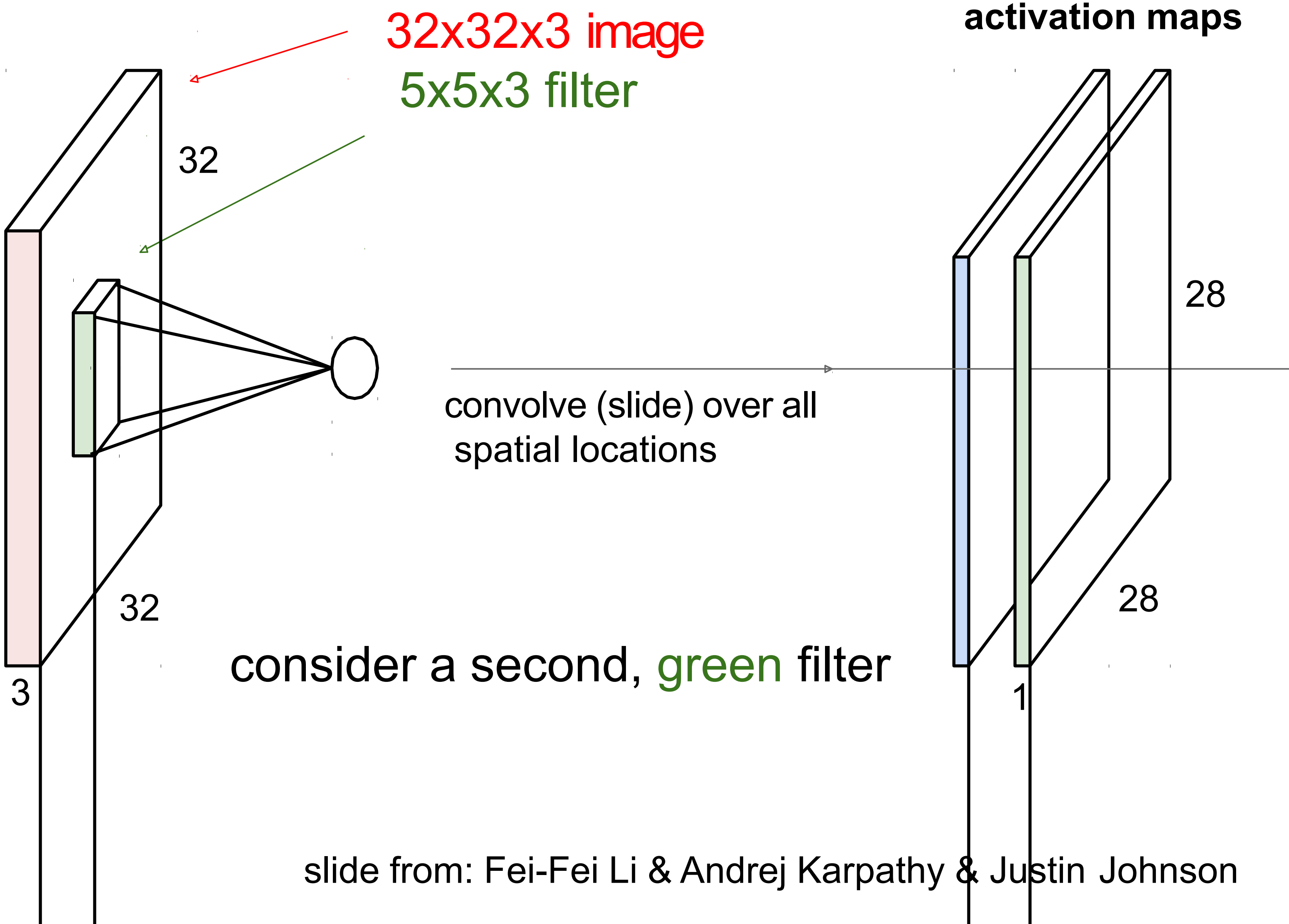
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

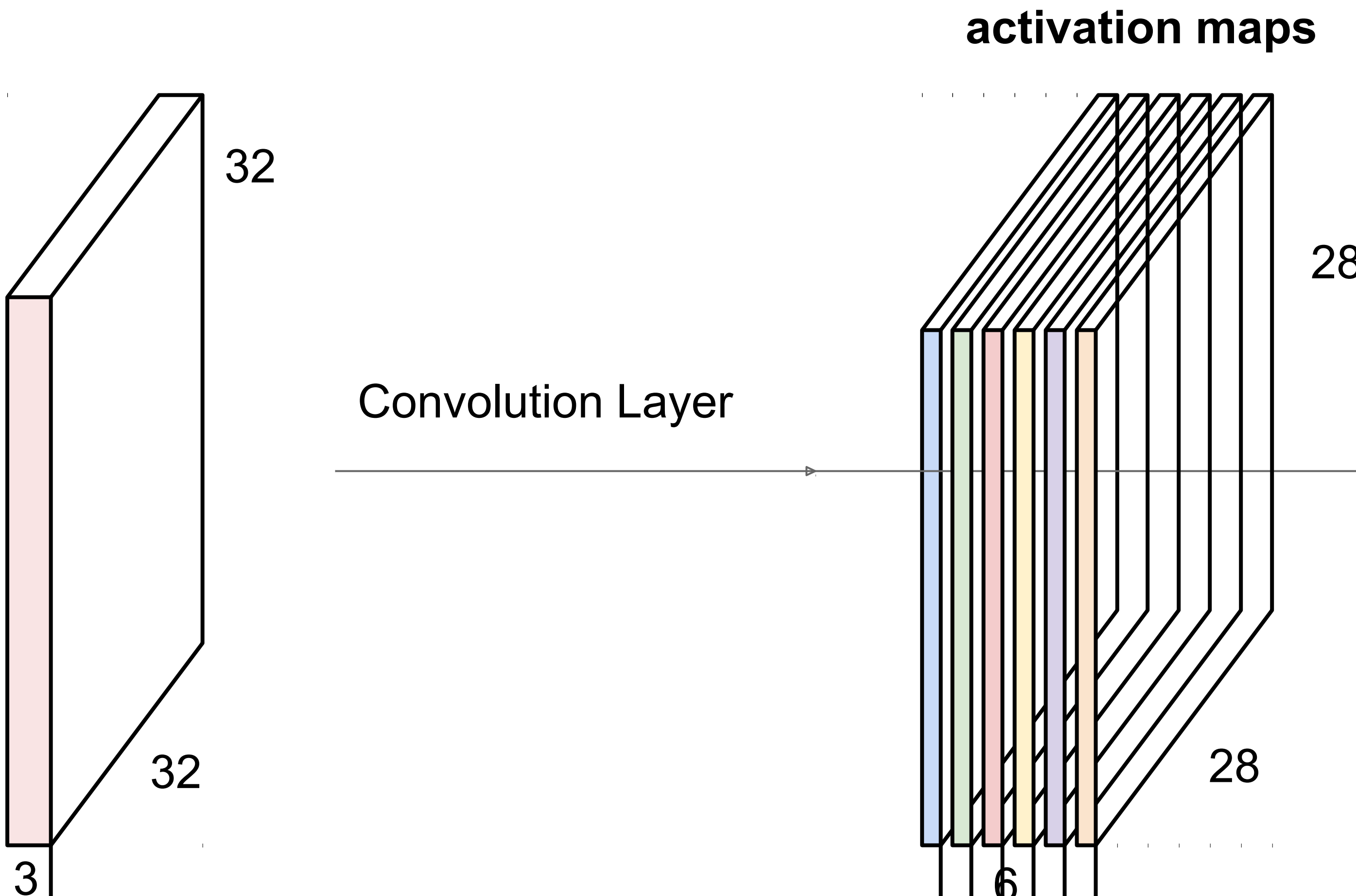
Convolution Layer



consider a second, green filter

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

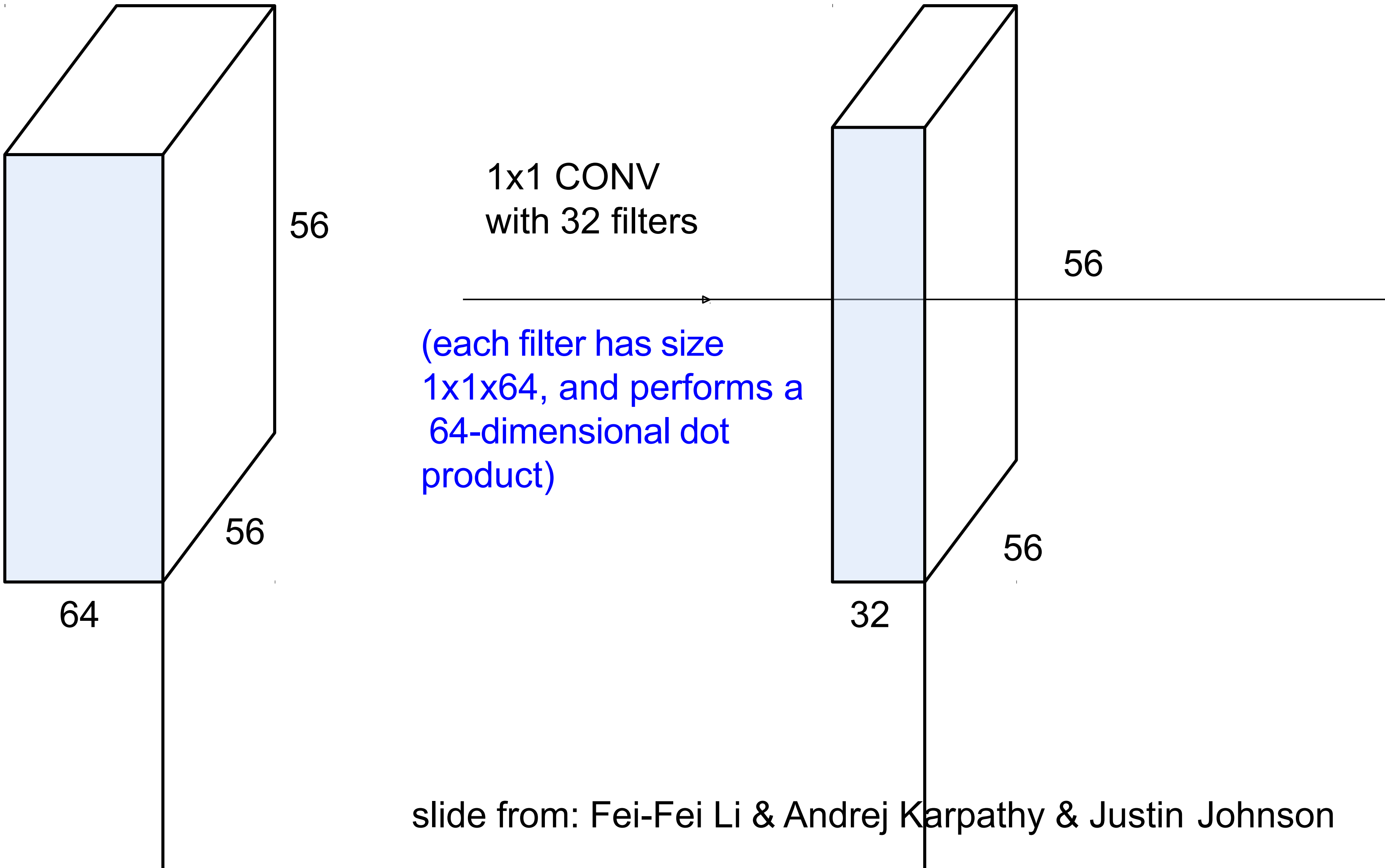
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

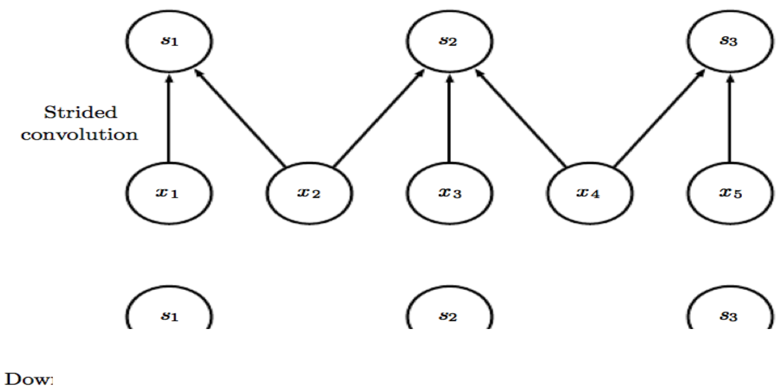
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution with 1x1 filters makes perfect sense

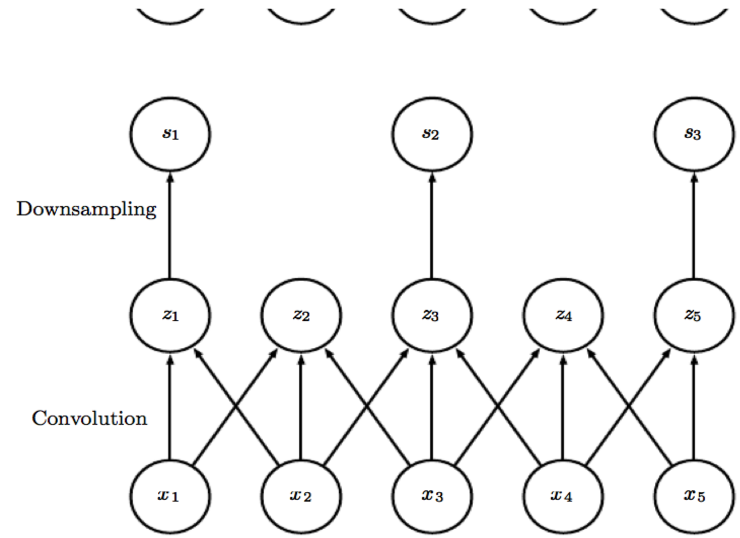


slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Stride

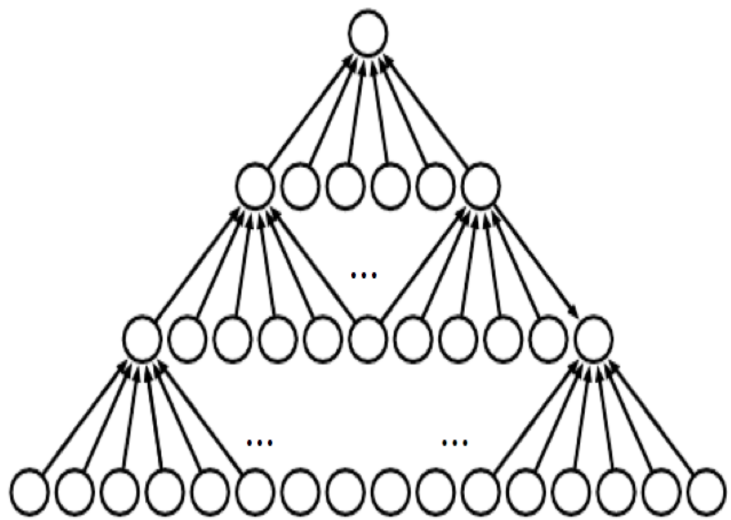


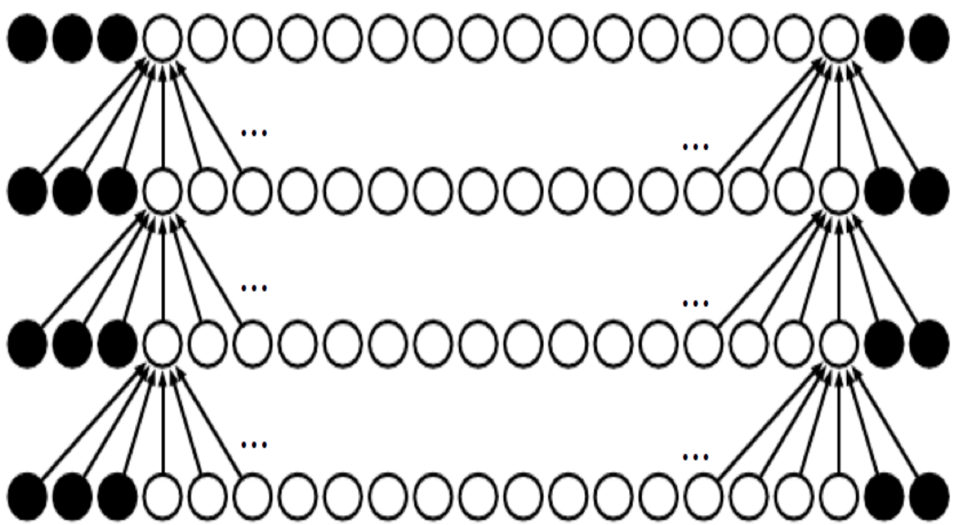
s_1
s_2
s_3
Strided convolution
x_1
x_2
x_3
x_4
x_5
s_1
s_2
s_3



s_1
s_2
s_3
Downsampling
z_1
z_2
z_3
z_4
z_5
Convolution
x_1
x_2
x_3
x_4
x_5

(Zero)-Padding

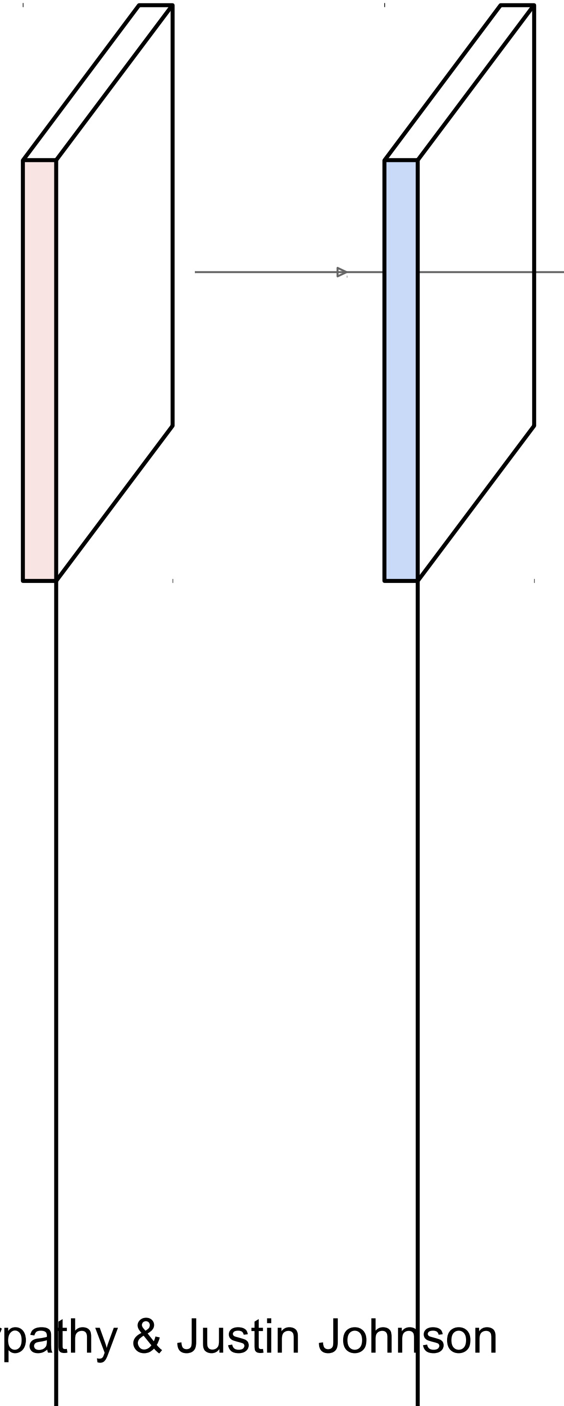




Example:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

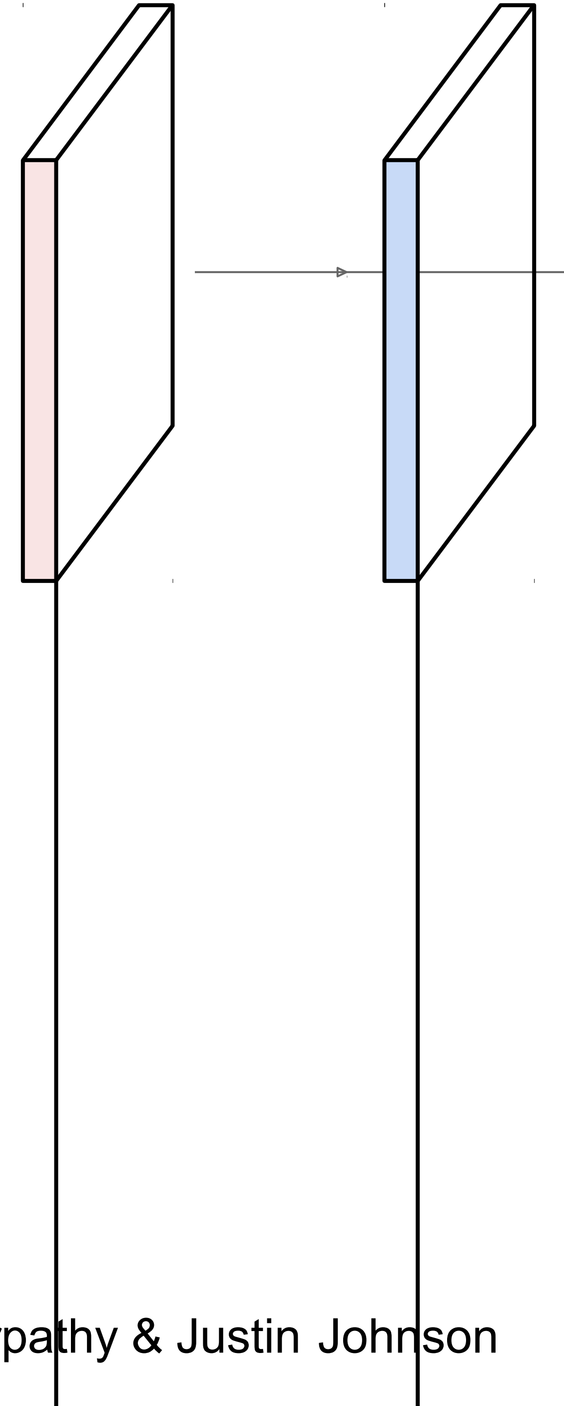


slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Example:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

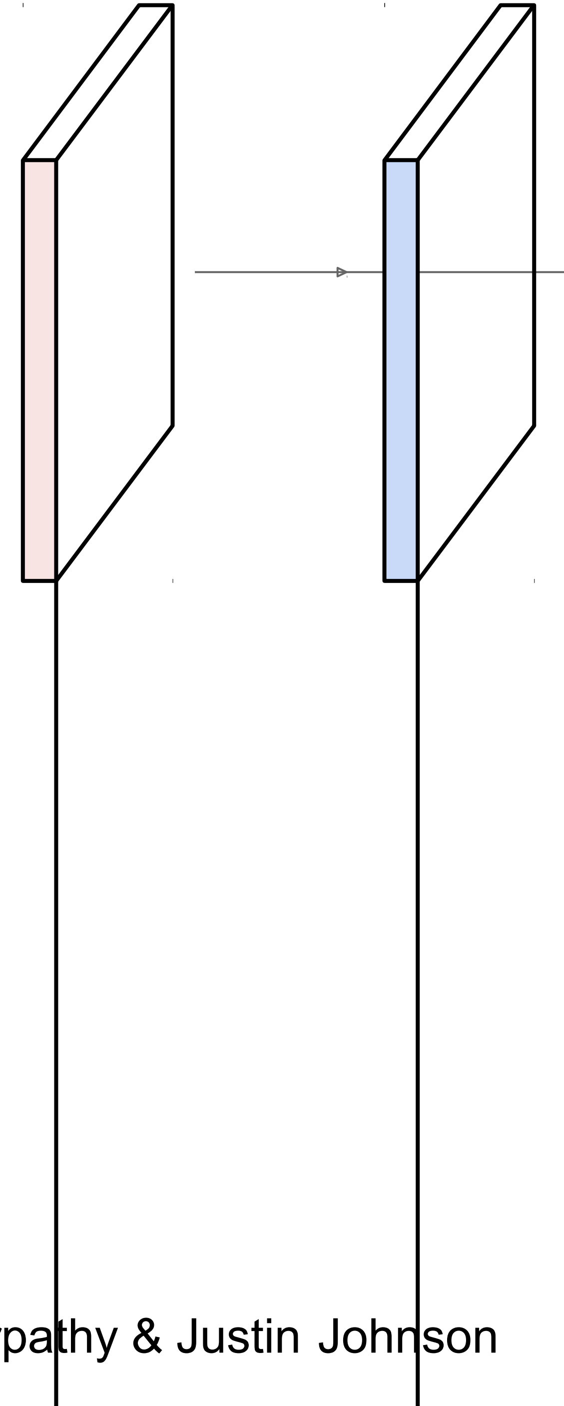


slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Example:

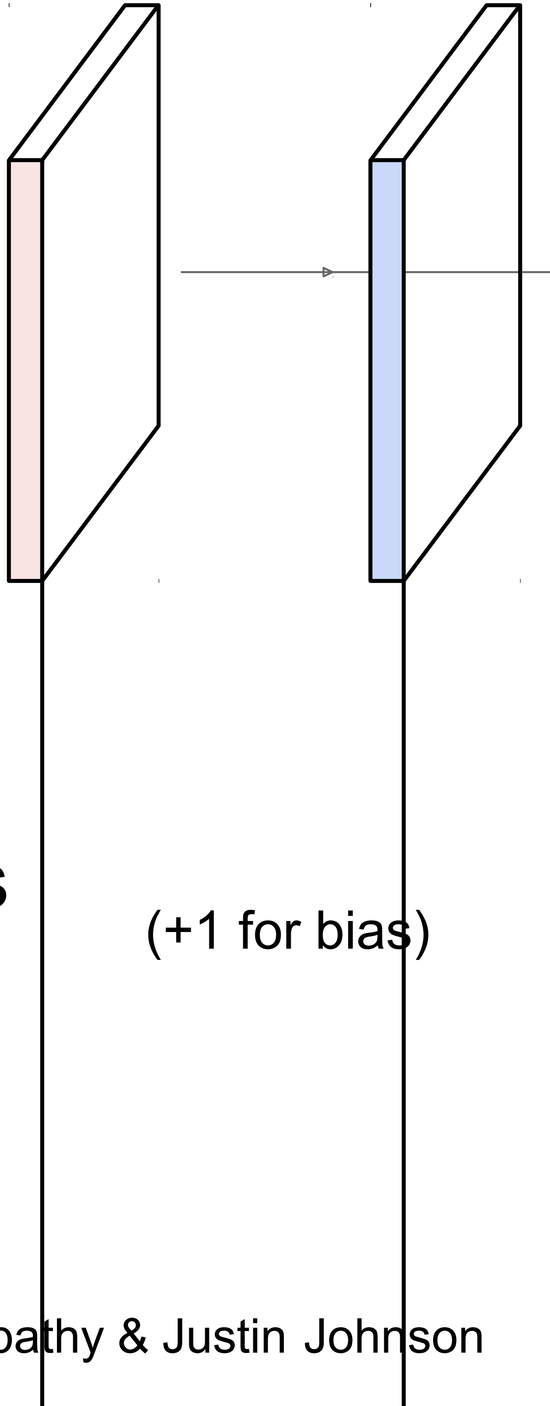
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Example:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

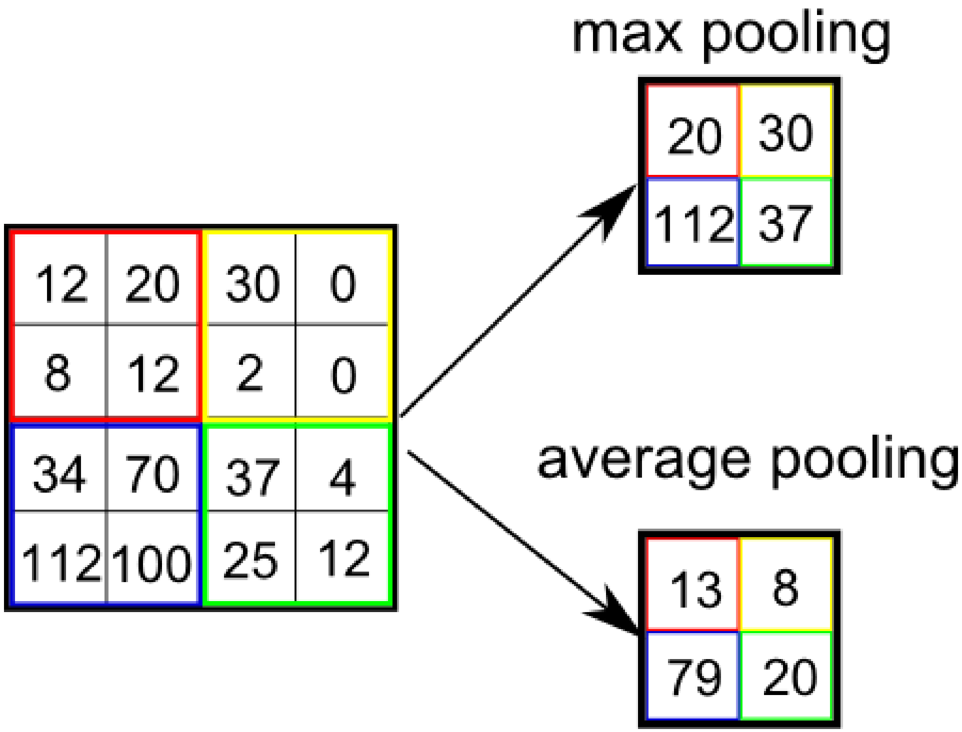
each filter has 5*5*3 + 1 = 76 params

(+1 for bias)

=> 76*10 = **760**

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

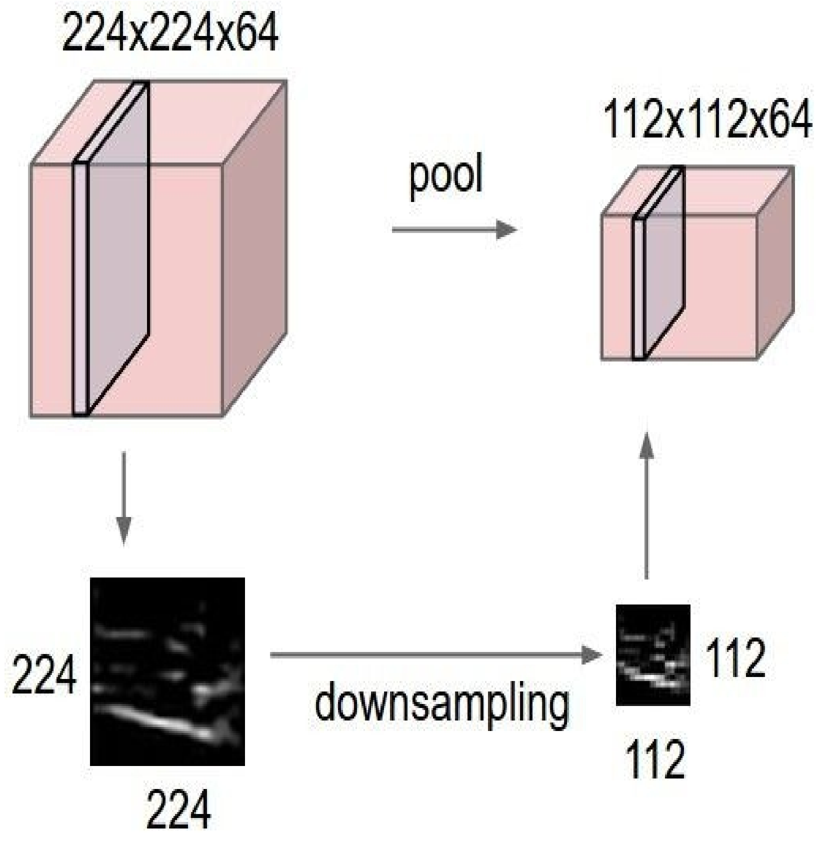
Pooling



Effect = invariance to small translations of the input

Pooling

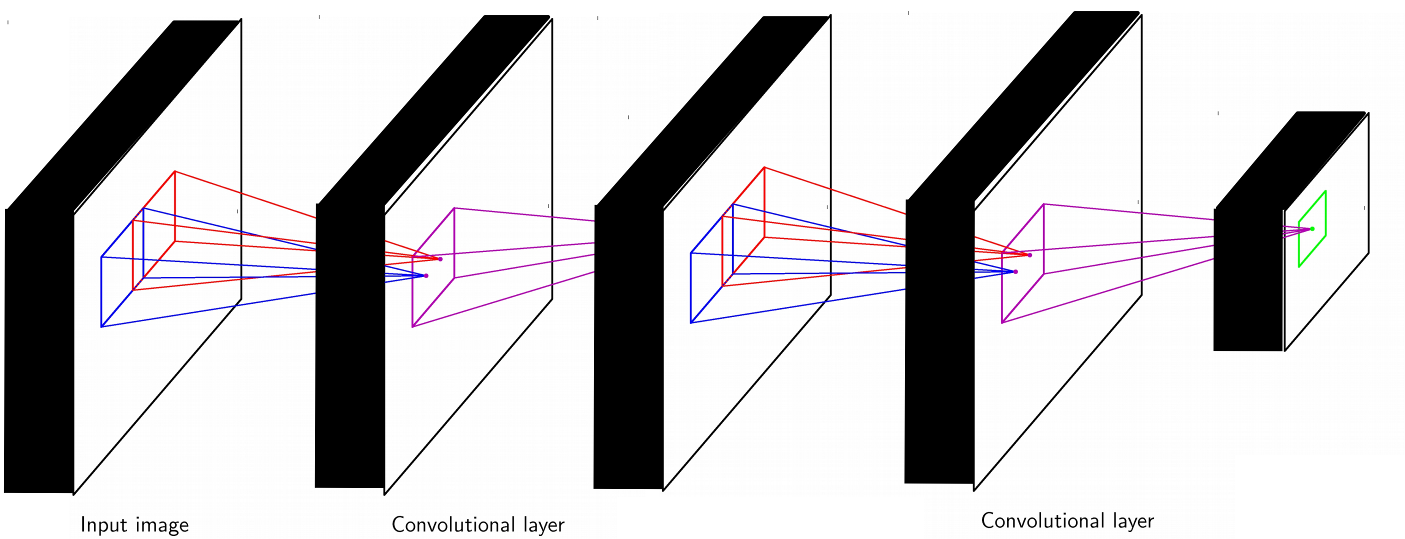
- makes the representations smaller and computationally less expensive
- operates over each activation map independently



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

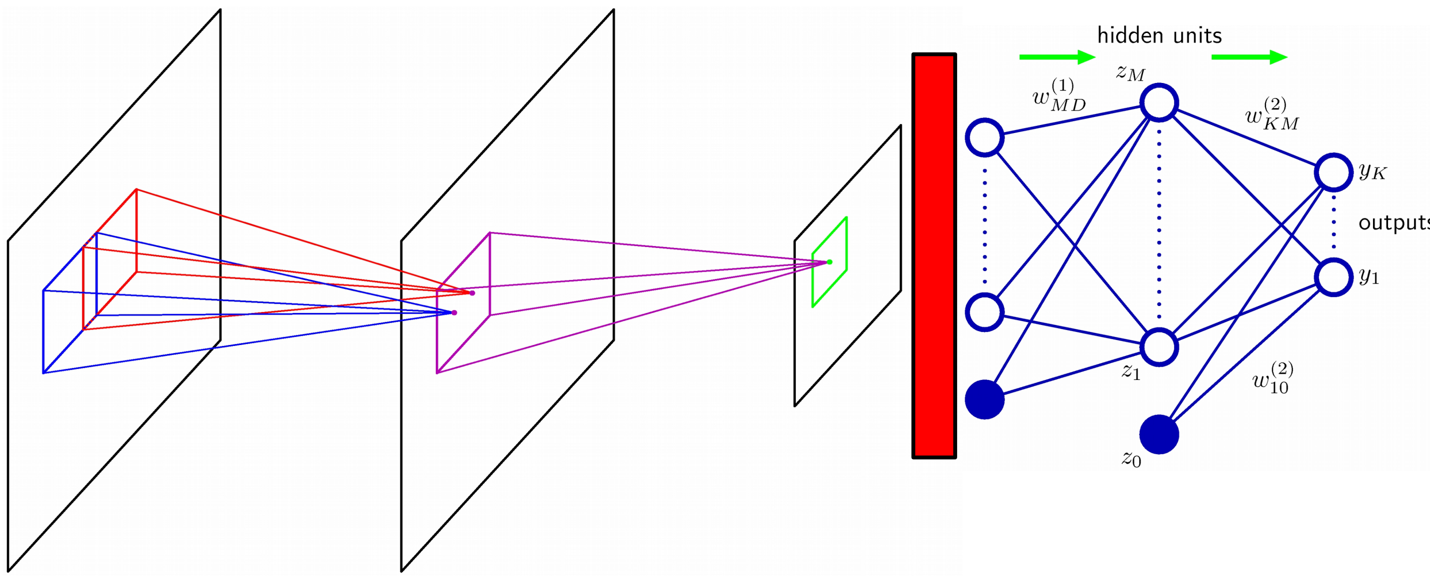
Receptive fields

- "Receptive field" is area in original image impacting a certain unit
  - Later layers can capture more complex patterns over larger areas
- Receptive field size grows linearly over convolutional layers
  - If we use a convolutional filter of size w x w, then each layer the receptive field increases by (w-1)
- Receptive field size increases exponentially over pooling layers
  - It is the stride that makes the difference, not pooling vs convolution



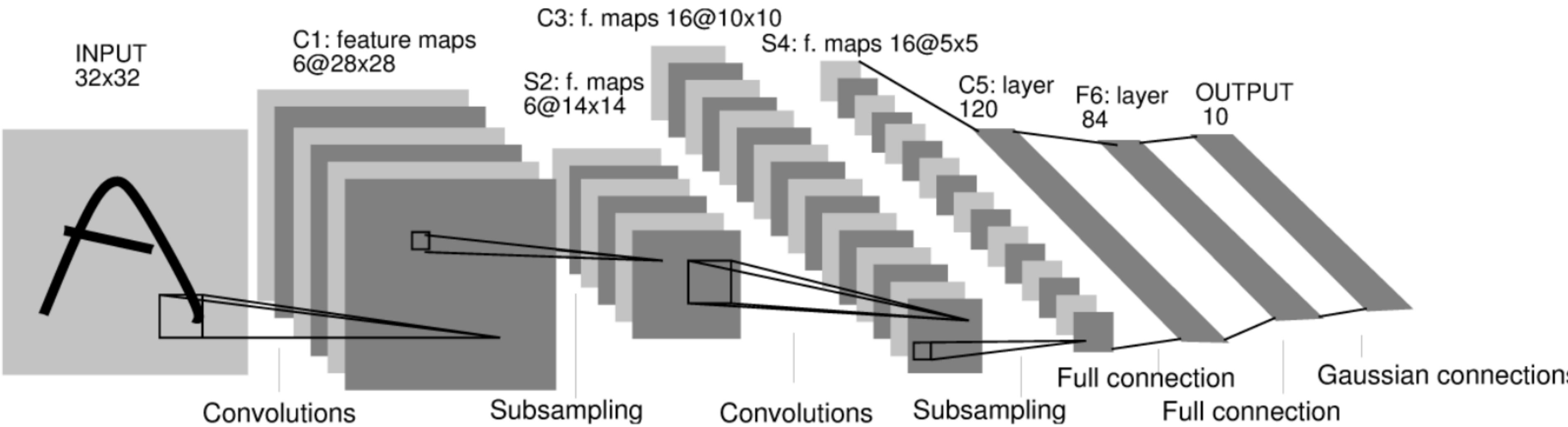
Fully connected layers

- Convolutional and pooling layers typically followed by several “fully connected” (FC) layers, i.e. standard multi-layer network
  - FC layer connects all units in previous layer to all units in next layer
  - Assembles all local information into global vectorial representation
- FC layers followed by softmax over outputs to generate distribution over image class labels
- First FC layer that connects response map to vector has many parameters
  - Conv layer of size 16x16x256 with following FC layer with 4096 units leads to a connection with 256 million parameters !



Convolutional neural network architectures

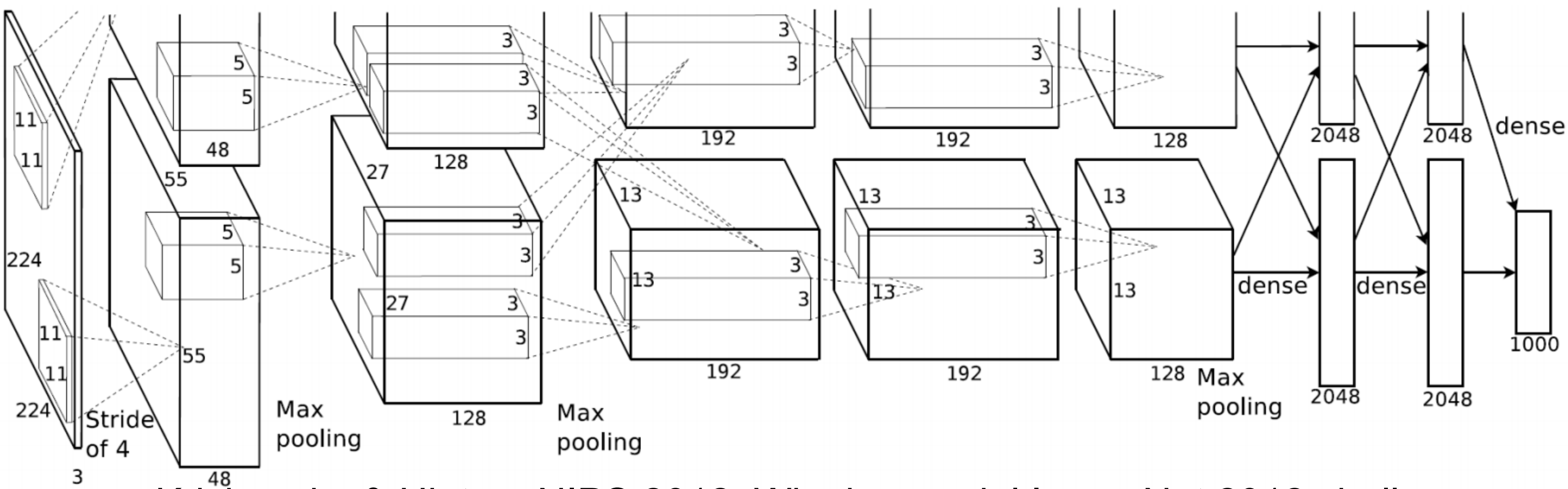
- Surprisingly little difference between todays architectures and those of late eighties and nineties
  - Convolutional layers, same
  - Nonlinearities: ReLU dominant now, tanh before
  - Subsampling: more strided convolution now than max/average pooling



Handwritten digit recognition network. LeCun, Bottou, Bengio, Haffner, Proceedings IEEE, 1998

Convolutional neural network architectures

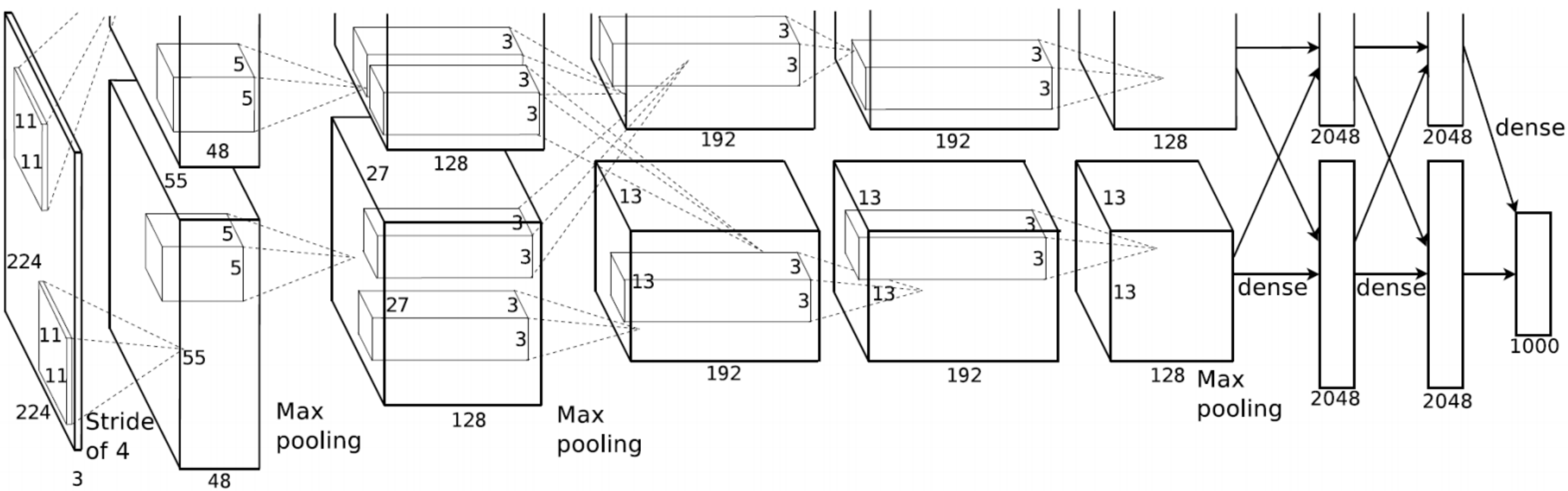
- Recent success with deeper networks
  - 19 layers in Simonyan & Zisserman, ICLR 2015
  - Hundreds of layers in residual networks, He et al. ECCV 2016
- More filters per layer: hundreds to thousands instead of tens
- More parameters: tens or hundreds of millions



Krizhevsky & Hinton, NIPS 2012, Winning model ImageNet 2012 challenge

Other factors that matter

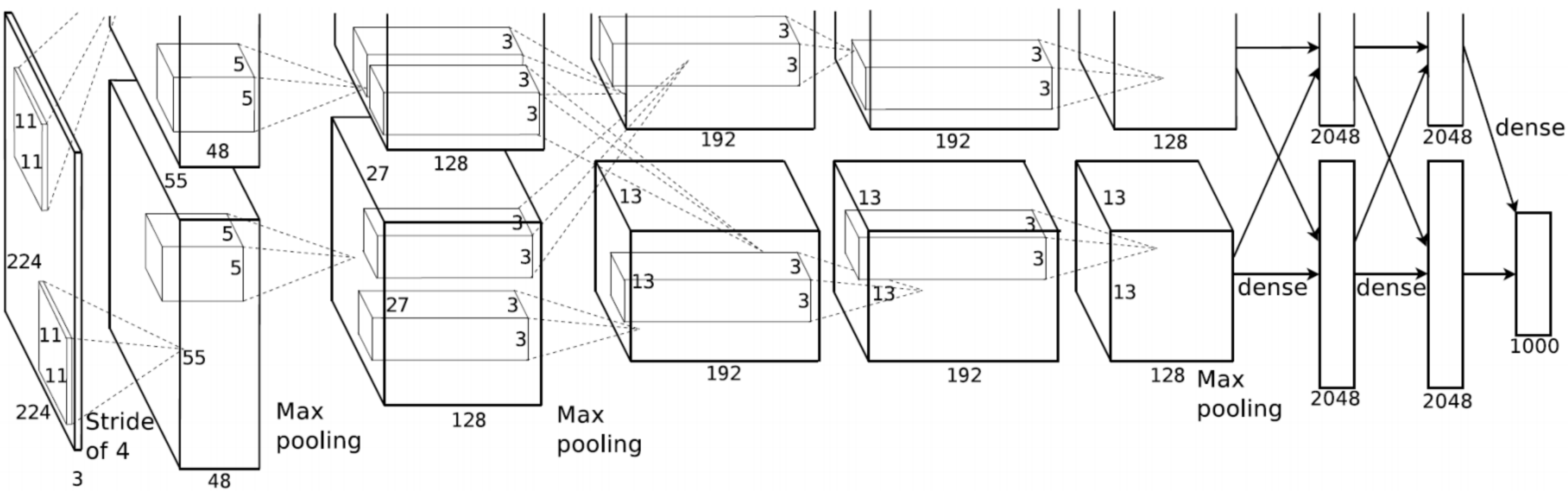
- More training data
  - 1.2 millions of 1000 classes in ImageNet challenge
  - 200 million faces in Schroff et al, CVPR 2015
- GPU-based implementations
  - Massively parallel computation of convolutions
  - Krizhevsky & Hinton, 2012: six days of training on two GPUs
  - Rapid progress in GPU compute performance



Krizhevsky & Hinton, NIPS 2012, Winning model ImageNet 2012 challenge

Understanding convolutional neural network activations

- Architecture consists of
  - 5 convolutional layers
  - 2 fully connected layers
- Visualization of patches that yield maximum response for certain units
  - We will look at each of the 5 convolutional layers

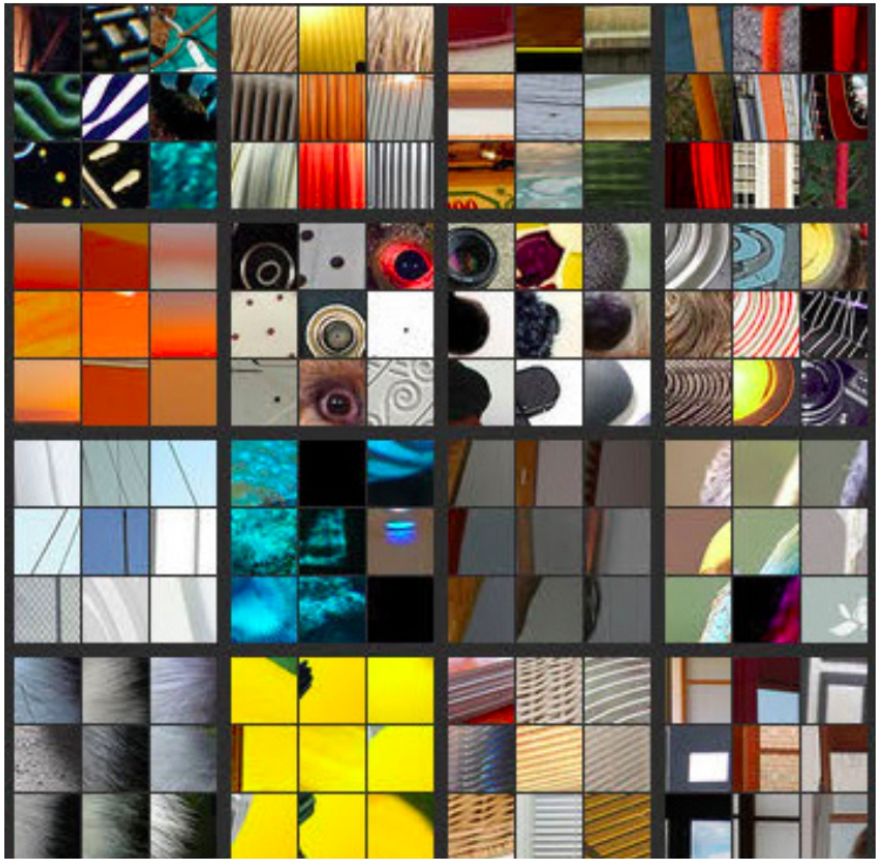


Krizhevsky & Hinton, NIPS 2012, Winning model ImageNet 2012 challenge

Understanding convolutional neural network activations

- Patches generating highest response for a selection of convolutional filters,
  - Showing 9 patches per filter
  - Zeiler and Fergus, ECCV 2014
- Layer 1: simple edges and color detectors

- Layer 2: corners, center-surround, ...



Understanding convolutional neural network activations

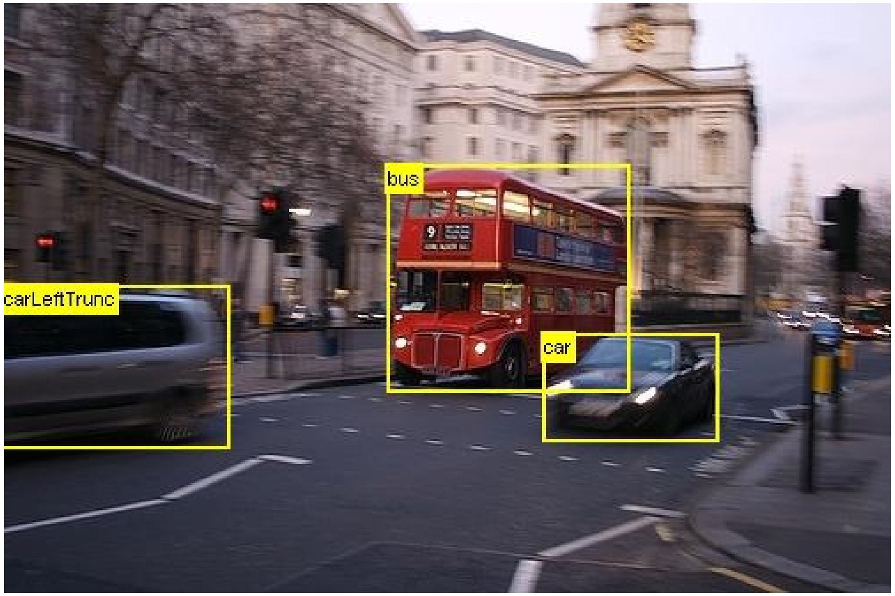
- Layer 3: various object parts

Understanding convolutional neural network activations

- Layer 4+5: selective units for entire objects or large parts of them

Convolutional neural networks for other tasks

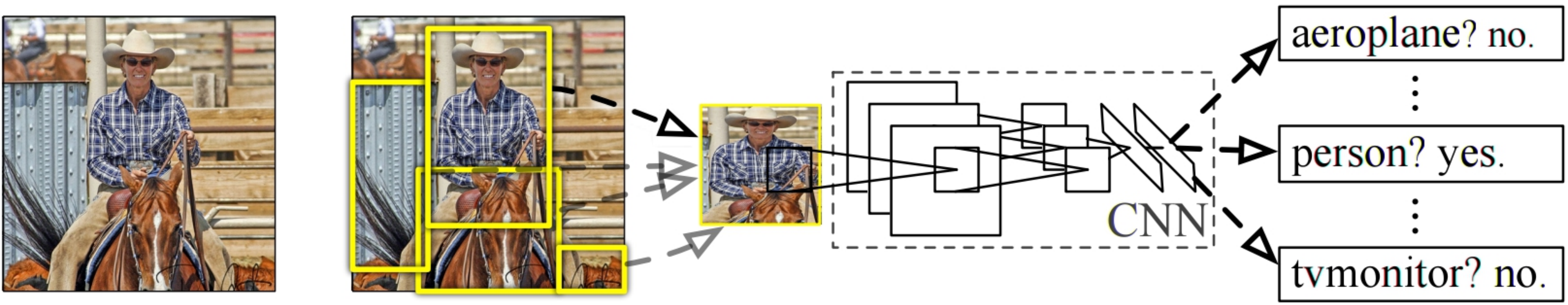
- **Object category localization**



- Semantic segmentation

CNNs for object category localization

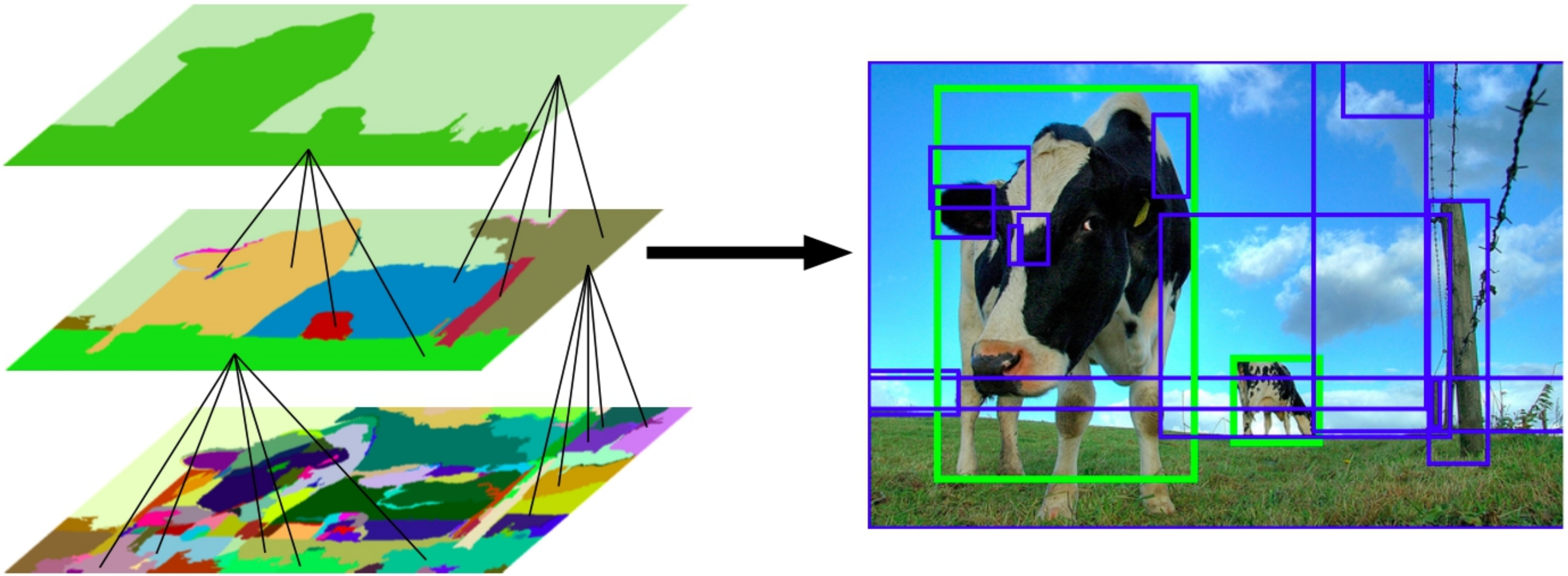
- Apply CNN image classification model to image sub-windows
  - For each window decide if it represents a car, sheep, ...
- Resize detection windows to fit CNN input size
- Unreasonably many image regions to consider if applied in naive manner
  - Use detection proposals based on low-level image contours



R-CNN, Girshick et al., CVPR 2014

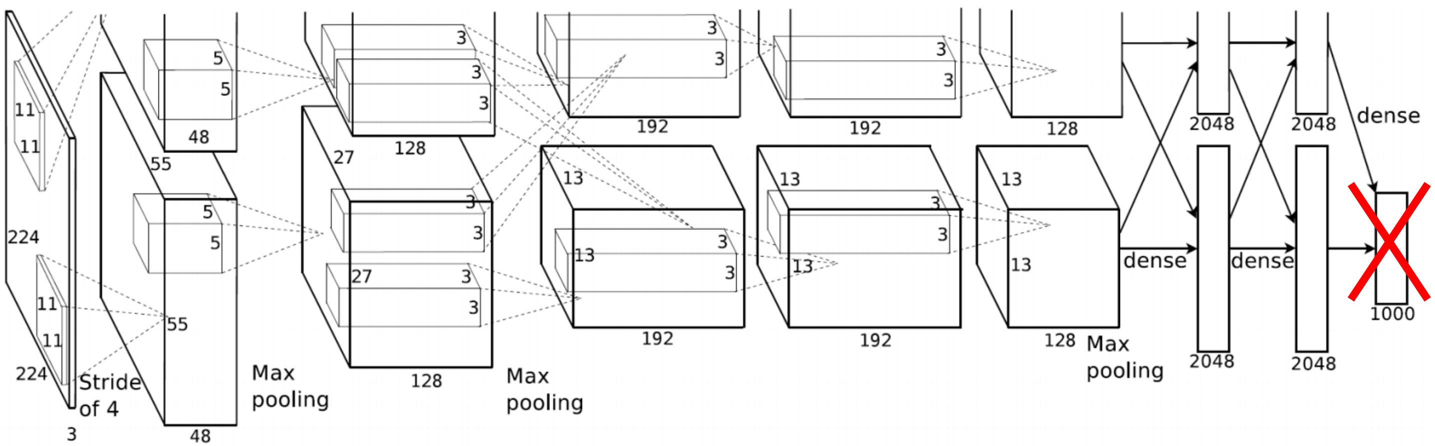
Detection proposal methods

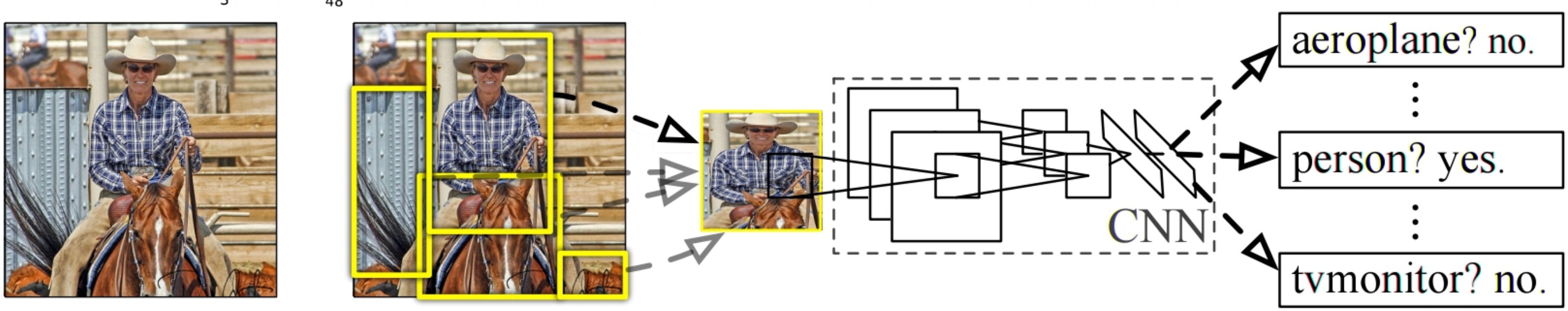
- Many methods exist, some based on learning others not
- Selective search method [Uijlings et al., IJCV, 2013]
  - Unsupervised multi-resolution hierarchical segmentation
  - Detections proposals generated as bounding box of segments
  - 1500 windows per image suffice to cover over 95% of true objects with sufficient accuracy



CNNs for object category localization

- On some datasets too little training data to learn CNN from scratch
  - Only few hundred objects instances labeled with bounding box
  - **Pre-train** AlexNet on large ImageNet classification problem
  - Replace last classification layer with classification over N categories + background
  - **Fine-tune** CNN weights for classification of detection proposals





CNNs for object category localization

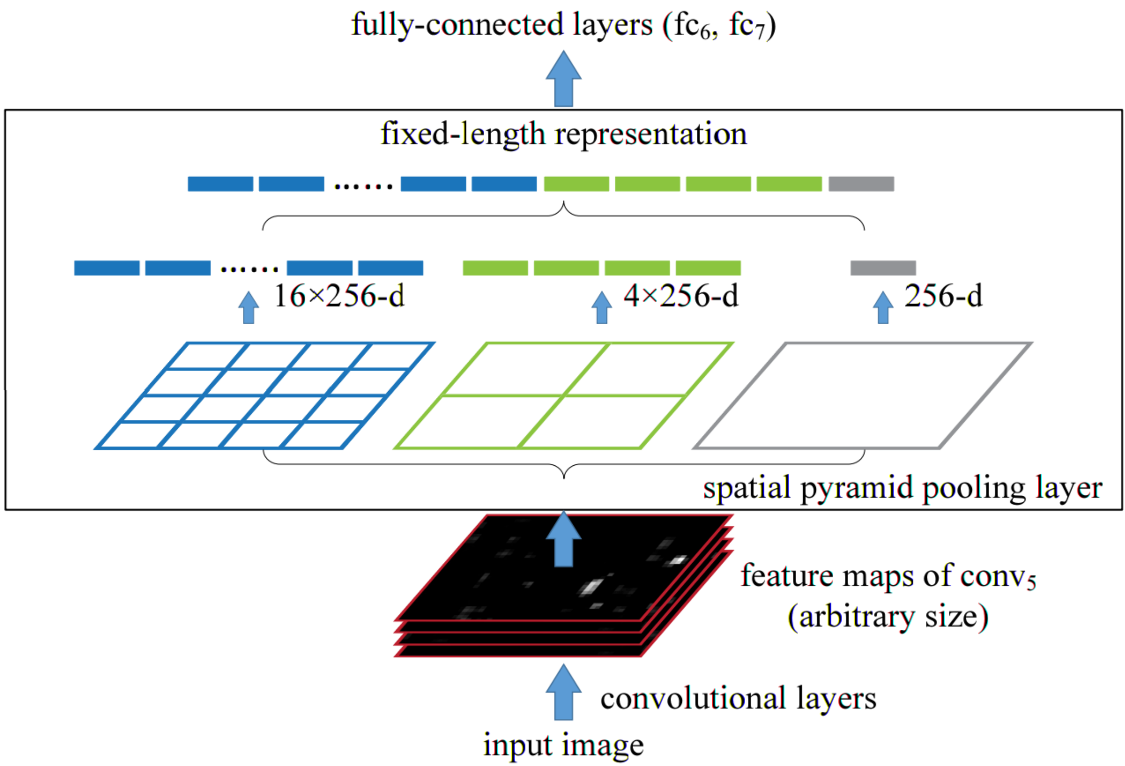
- Comparison with state of the art non-CNN models
  - Object detection is correct if window has intersection/union with ground-truth window of at least 50%
- Significant increase in performance of 10 points mean-average-precision (mAP)

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [20]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

Table 1: Detection average precision (%) on VOC 2010 test. R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression (BB) is described in Section C. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. †DPM and SegDPM use context rescoring not used by the other methods.

Efficient object category localization with CNN

- R-CNN recomputes convolutions many times across overlapping regions
- Instead: compute convolutional part only once across entire image
- For each window:
  - Pool convolutional features using max-pooling into fixed-size representation
  - Fully connected layers up to classification computed per window



SPP-net, He et al., ECCV 2014

Efficient object category localization with CNN

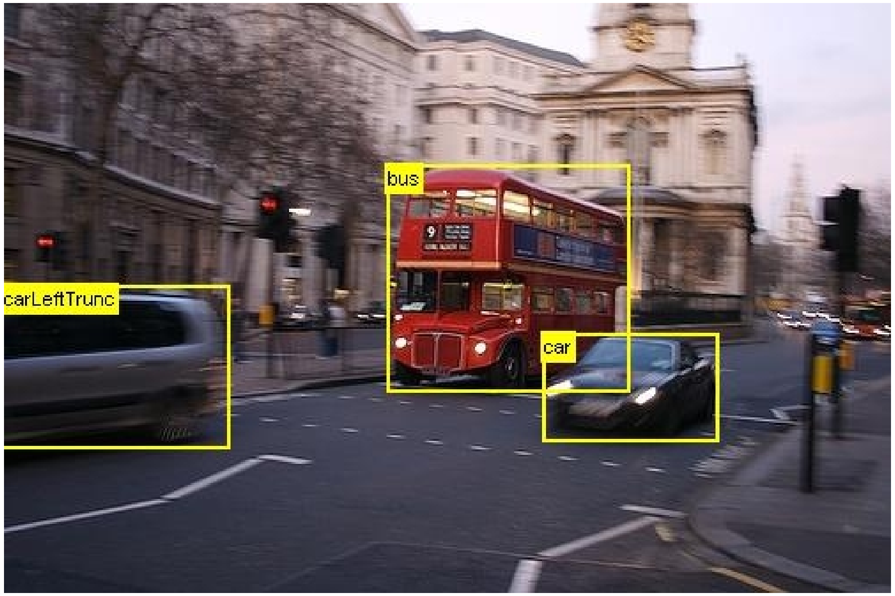
- Refinement: Compute convolutional filters at multiple scales
  - For given window use scale at which window has roughly size 224x224
- Similar performance as explicit window rescaling, and re-computing convolutional filters
- Speedup of about 2 orders of magnitude

	SPP (1-sc) (ZF-5)	SPP (5-sc) (ZF-5)	R-CNN (ZF-5)
ftfc$_7$	54.5	<u>55.2</u>	55.1
ftfc$_7$ bb	58.0	**59.2**	**59.2**
conv time (GPU)	0.053s	0.293s	14.37s
fc time (GPU)	0.089s	0.089s	0.089s
total time (GPU)	0.142s	0.382s	14.46s
speedup (*vs.* RCNN)	**102**$\times$	**38**$\times$	-

Table 10: Detection results (mAP) on Pascal VOC 2007, **using the same pre-trained model** of SPP (ZF-5).

Convolutional neural networks for other tasks

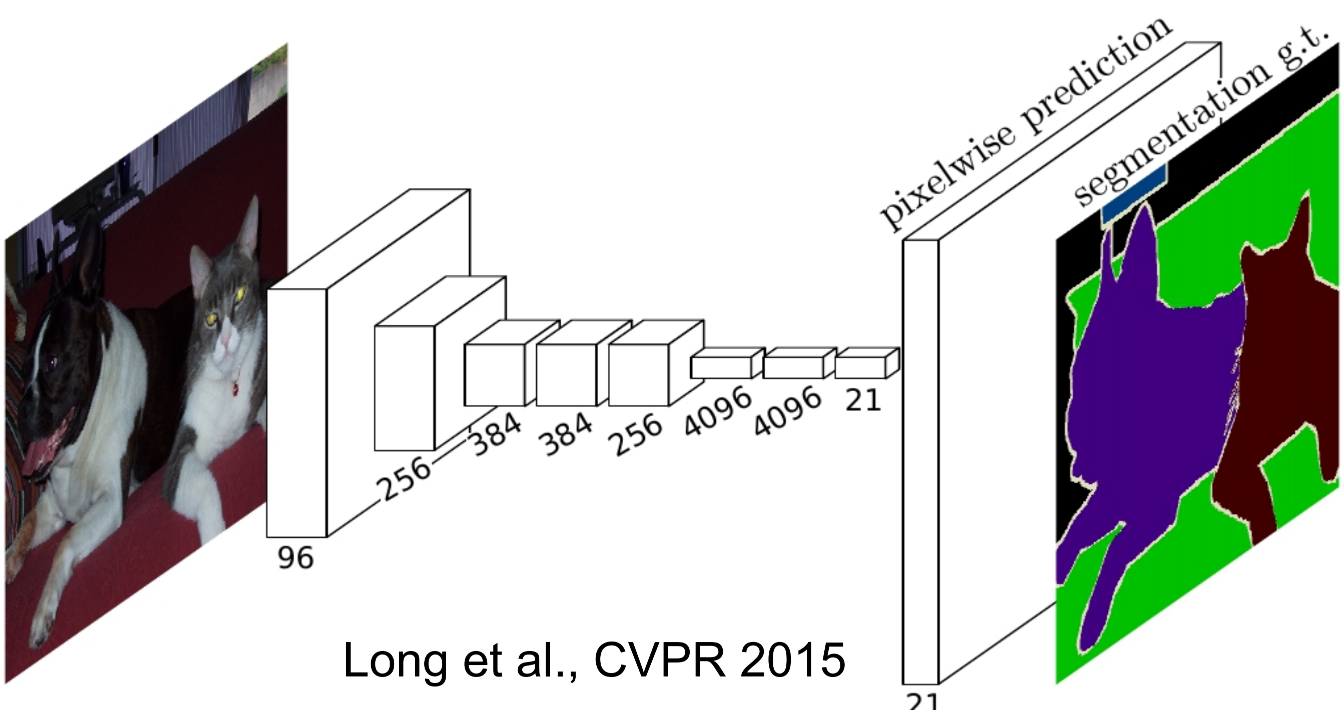
- Object category localization



- **Semantic segmentation**

Application to semantic segmentation

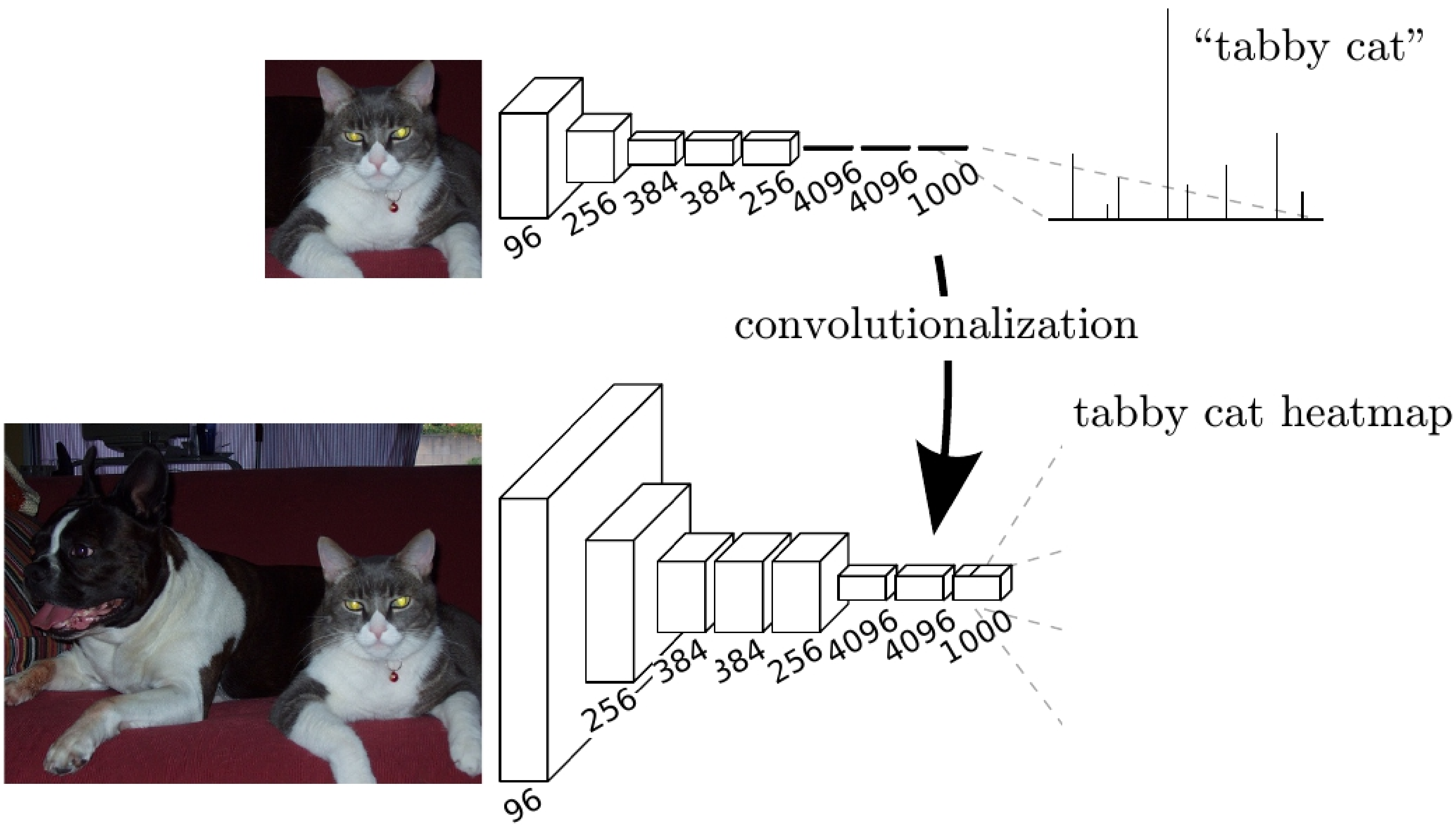
- Assign each pixel to an object or background category
  - Consider running CNN on small image patch to determine its category
  - Train by optimizing per-pixel classification loss
- Similar to SPP-net: want to avoid wasteful computation of convolutional filters
  - Compute convolutional layers once per image
  - Here all local image patches are at the same scale
  - Many more local regions: dense, at every pixel



Long et al., CVPR 2015

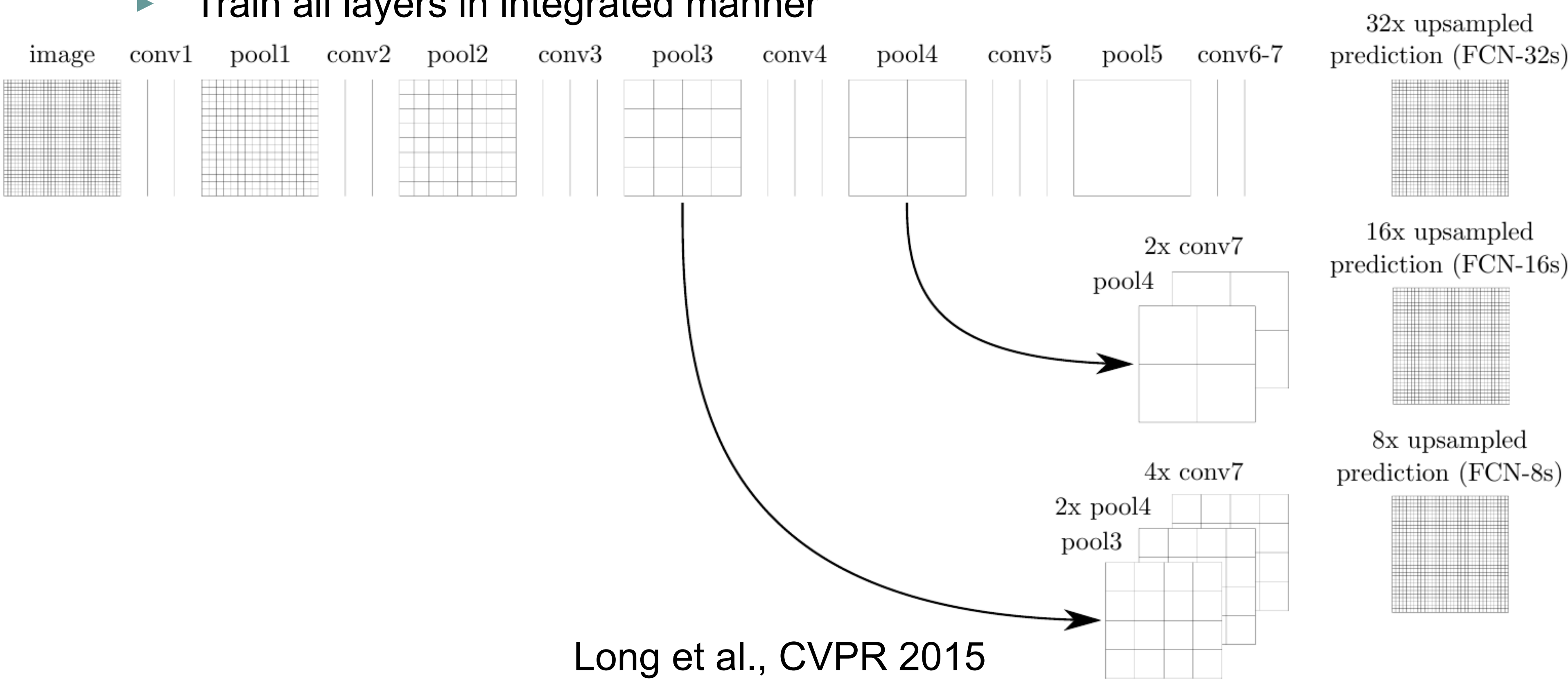
Application to semantic segmentation

- Interpret fully connected layers as 1x1 sized convolutions
  - Function of features in previous layer, but only at own position
  - Still same function is applied at all positions
- Five sub-sampling layers reduce the resolution of output map by factor 32



Application to semantic segmentation

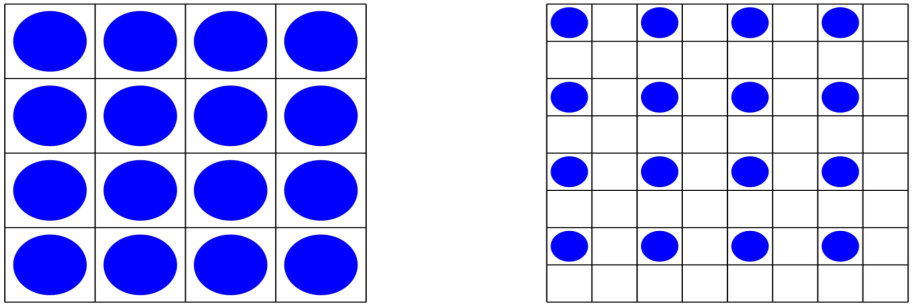
- Idea 1: up-sampling via bi-linear interpolation
  - Gives blurry predictions
- Idea 2: weighted sum of response maps at different resolutions
  - Upsampling of the later and coarser layer
  - Concatenate fine layers and upsampled coarser ones for prediction
  - Train all layers in integrated manner



Long et al., CVPR 2015

Upsampling of coarse activation maps

- Simplest form: use bilinear interpolation or nearest neighbor interpolation
  - Note that these can be seen as upsampling by zero-padding, followed by convolution with specific filters, no channel interactions
- Idea can be generalized by learning the convolutional filter
  - No need to hand-pick the interpolation scheme
  - Can include channel interactions, if those turn out be useful

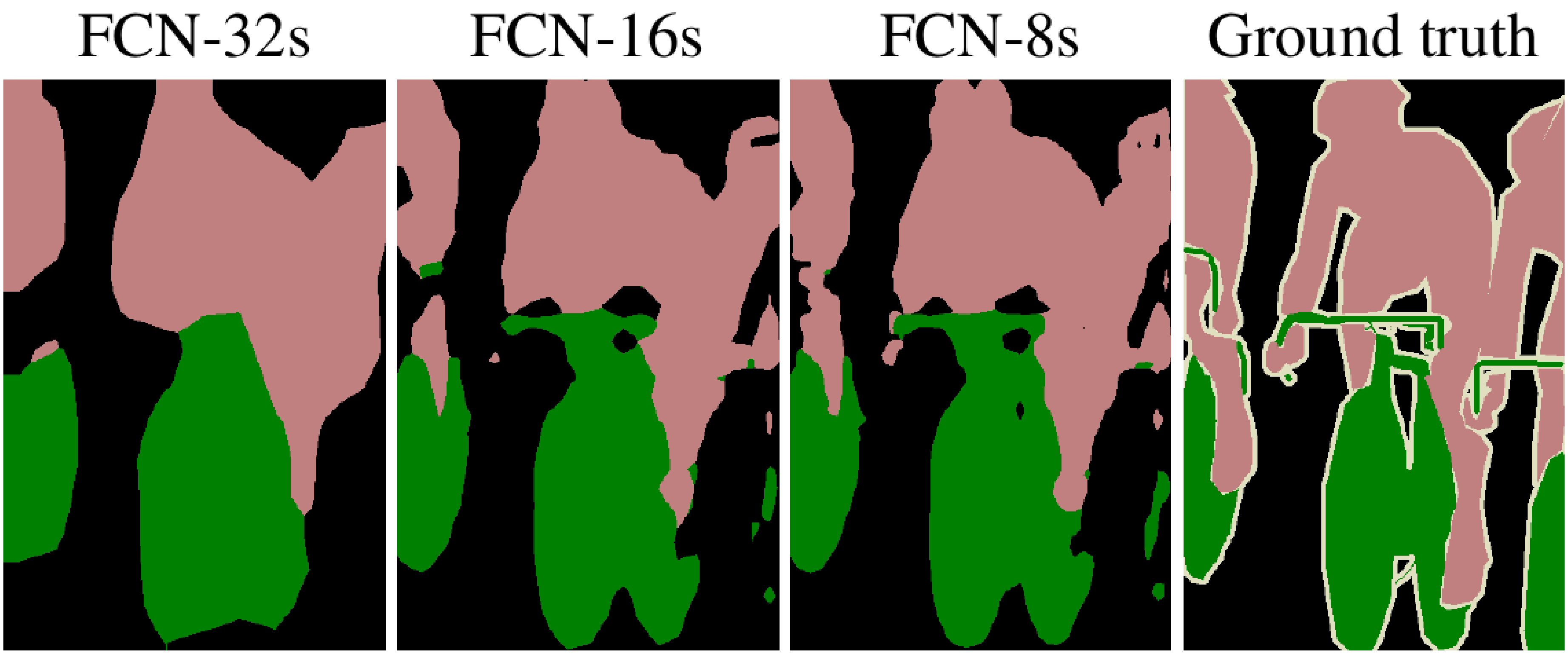


Bi-linear: $\frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$ Nearest neighbor: $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

- Resolution-increasing counterpart of strided convolution
  - Average and max pooling can be written in terms of convolutions
  - See: “Convolutional Neural Fabrics”, Saxena & Verbeek, NIPS 2016.

Application to semantic segmentation

- Results obtained at different resolutions
  - Detail better preserved at finer resolutions



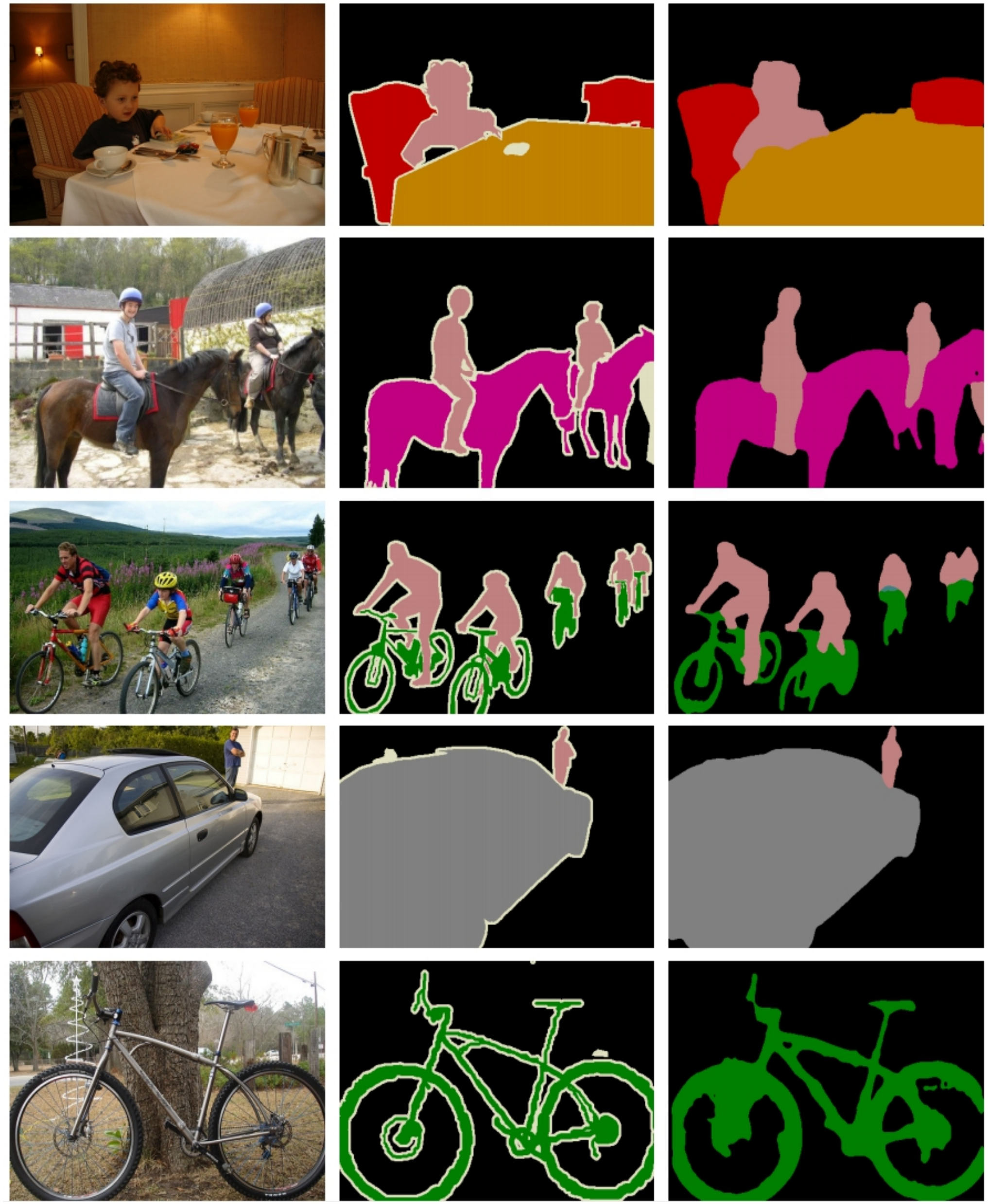
Semantic segmentation: further improvements

- Beyond independent prediction of pixel labels
  - Integrate conditional random field (CRF) models with CNN

    Zheng et al., ICCV'15

- Using more sophisticated upsampling schemes to maintain high-resolution signals

  Lin et al., arXiv 2016



Summary feed-forward neural networks

- Construction of complex functions with circuits of simple building blocks
  - Linear function of previous layers
  - Scalar non-linearity
- Learning via back-propagation of error gradient throughout network
  - Need directed acyclic graph
- Convolutional neural networks (CNNs) extremely useful for image data
  - State-of-the-art results in a wide variety of computer vision tasks
  - Spatial invariance of processing (also useful for video, audio, ...)
  - Stages of aggregation of local features into more complex patterns
  - Same weights shared for many units organized in response maps
- Applications for object localization and semantic segmentation
  - Local classification at level of detection windows or pixels
  - Computation of low-level convolutions can be shared across regions