Neural Network and Deep Learning

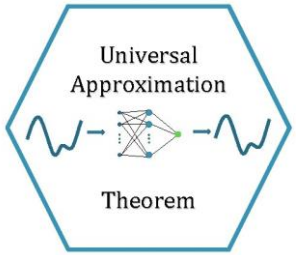
Lecture4:

Deep Learning
&
Convolutional Neural Networks (CNNs)

OUTLINE

- Deep Learning Fundamentals
- High-level features
- Deep projects
- Model Architectures
- Image Classification with Linear Classifier
- A bit of history
- Fast-forward: ConvNets are everywhere
- Convolutional Neural Networks (CNNs)

DEEP LEARNING FUNDAMENTALS & UNIVERSAL APPROXIMATION THEOREM



Question: What is the minimum number of hidden layers in a multilayer perceptron with an input–output mapping that provides an approximate realization of any continuous mapping?

Answer: The answer to this question is embodied in the universal approximation theorem for a nonlinear input–output mapping, which may be stated as follows:

Universal Approximation Theorem

Let $\varphi(.)$ *be a nonconstant, bounded, and monotone increasing continuous function and* $\varepsilon > 0$, *there exist an integer* m_1 *and real constants* $\alpha_i, b_i,$ *and* $w_{i,j}$, *where* $i = 1, \dots, m_1$ *and* $j = 1, \dots, m_0$ *such that we may define*

$$F(x_1, \dots, x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i\right)$$

as an approximation realization of function $f(.)$; *that is,*

$$\left|F(x_1, \dots, x_{m_0}) - f(x_1, \dots, x_{m_0})\right| < \varepsilon$$

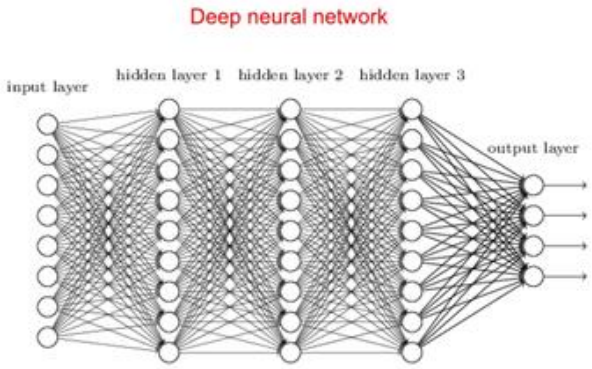
for all $x_1, x_2, \dots, x_{m_0}$ *that lie in the input space.*

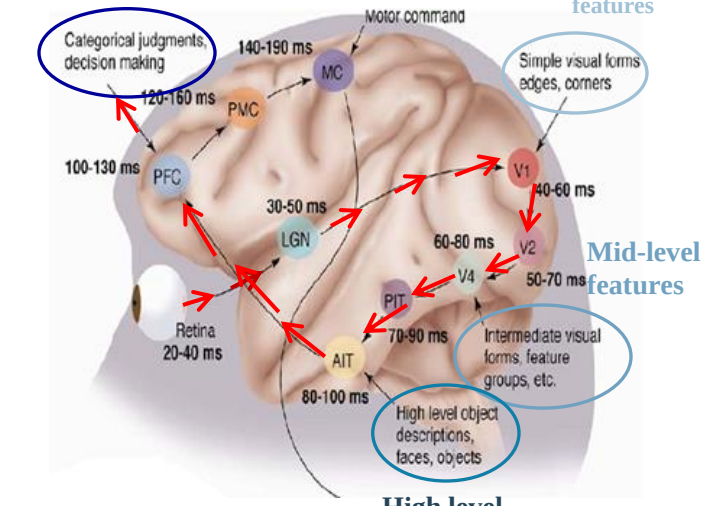
DEEP LEARNING FUNDAMENTALS

- 1-layer networks are too simple and can only work on linear problems, and we have introduced the Universal Approximation Theorem, showing how 2-layer neural networks with just one hidden layer are able to approximate to any degree any continuous function on a compact subset of R_n.
- **Deep neural networks**, that is, neural networks with at least two or more hidden layers.
- **Why deep learning?**
- In deep learning, the network does not simply learn to predict an output Y given an input X, but it also understands basic features of the input. In deep learning, the neural network is able to make abstractions of the features that comprise the input examples, to understand the basic characteristics of the examples, and to make predictions based on those characteristics. In deep learning, there is a level of abstraction that is missing in other basic machine learning algorithms or in shallow neural networks.

HIGH-LEVEL FEATURES

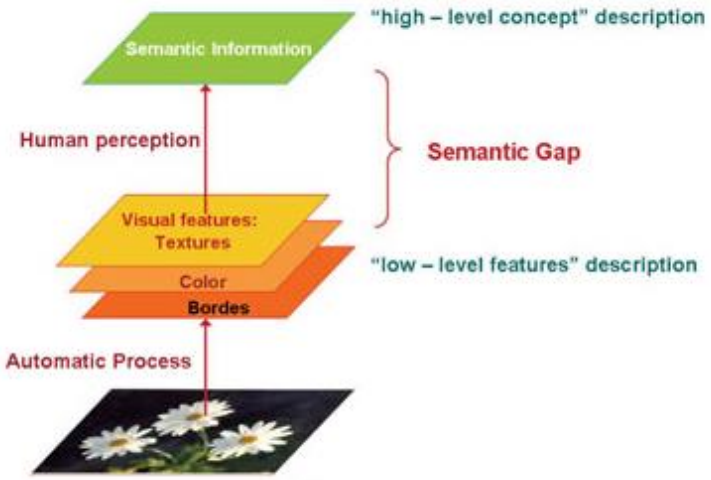
- The human brain is organized in a deep architectur
- The brain also appears to process information through multiple stages of transformation and representation.
- This is particularly clear in the primate visual system
- Visual system's sequence of processing stages: detection of
  - Edges
  - Primitive shapes
  - Moving up to gradually more complex visual shapes

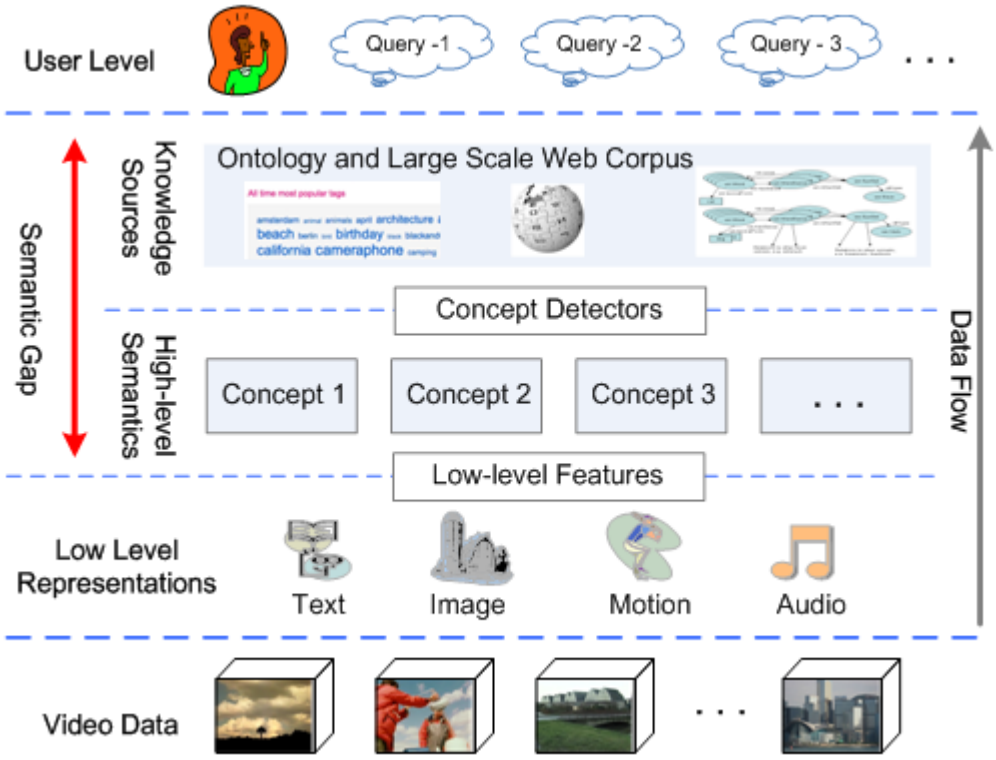




HIGH-LEVEL FEATURES

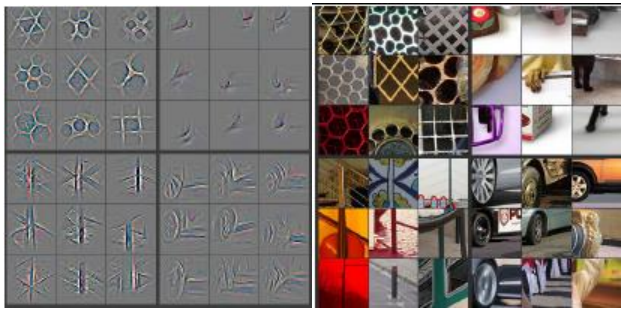
- **Semantic gap**



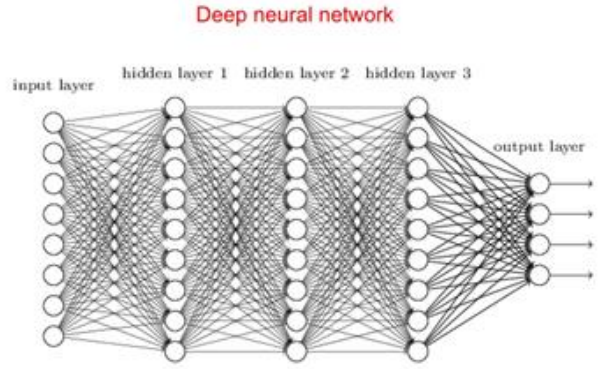


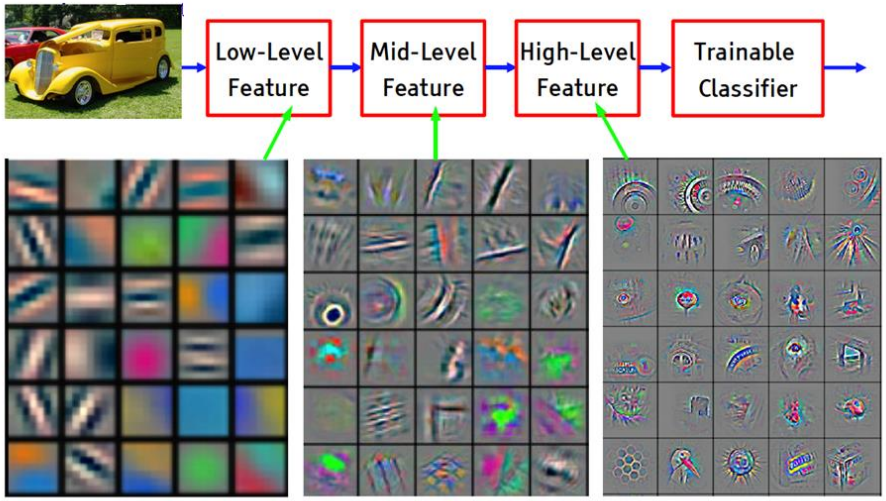
HIGH-LEVEL FEATURES

Layer 3



Layer 4

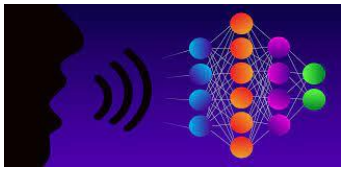




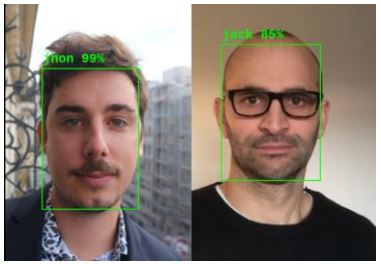
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

DEEP PROJECTS

Modern Art Generator



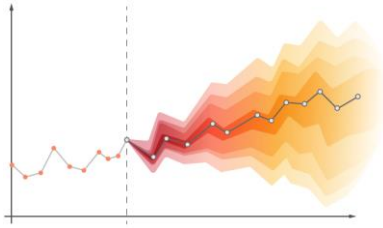
Speech Recognition



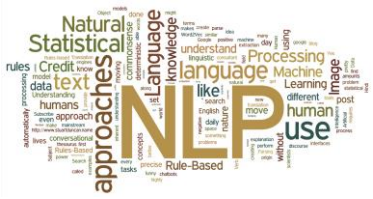
Face Recognition

Image Classification

Deepfake Videos

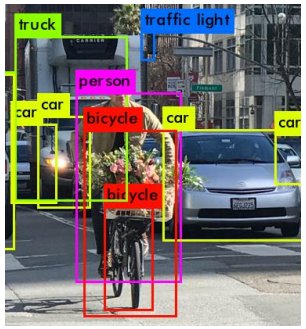


Time Series Forecasting



Natural Language Processing

Image Captioning



Object Detection

Machine Translation

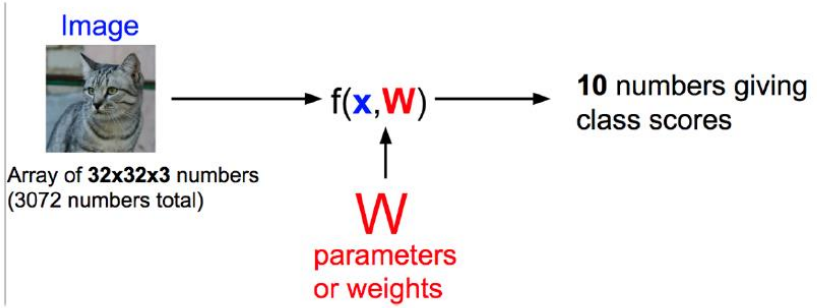
Segmentation

DeepDream

MODEL ARCHITECTURES

- Deep learning architectures refer to the various structured approaches that dictate how neurons and layers are organized and interact in neural networks. These architectures have evolved to tackle different problems and data types effectively.
  - Multi-Layer Perceptrons (MLPs)
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)
  - Generative Adversarial Networks (GANs)
  - Autoencoders
  - Transformer Networks
  - …

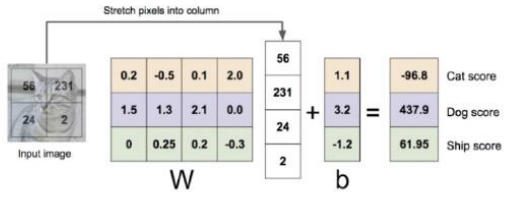
IMAGE CLASSIFICATION WITH LINEAR CLASSIFIER



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

Algebraic Viewpoint

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

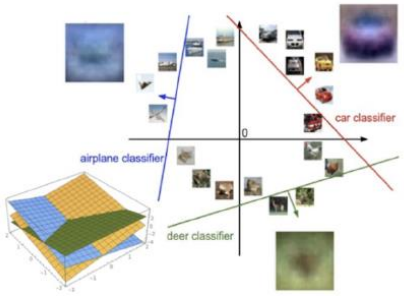


Visual Viewpoint

One template per class

Geometric Viewpoint

Hyperplanes cutting up space



Slide inspiration: Fei-Fei Li, Ehsan Adeli

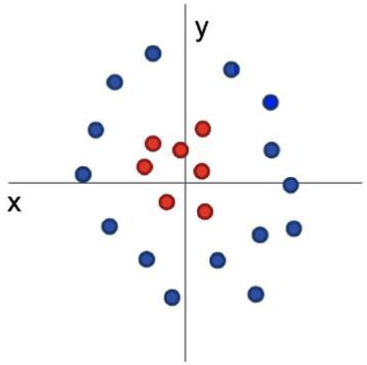
IMAGE CLASSIFICATION WITH LINEAR CLASSIFIER

Problem: Linear Classifiers are not very powerful

Visual Viewpoint

Linear classifiers learn one template per class

Geometric Viewpoint



Linear classifiers can only draw linear decision boundaries

Slide inspiration: Fei-Fei Li, Ehsan Adeli

A BIT OF HISTORY

Hubel & Wiesel,

1959

RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

J. Physiol. (1959) 148, 574–591

RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

By D. H. HUBEL* and T. N. WIESEL*
From the Wilmer Institute, The Johns Hopkins Hospital and University, Baltimore, Maryland, U.S.A.

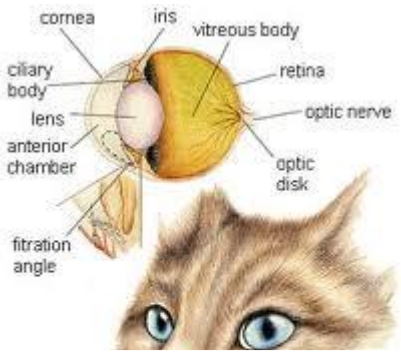
J. Physiol. (1962), **160**, *pp.* 106–154
With 2 plates and 20 text-figures
Printed in Great Britain

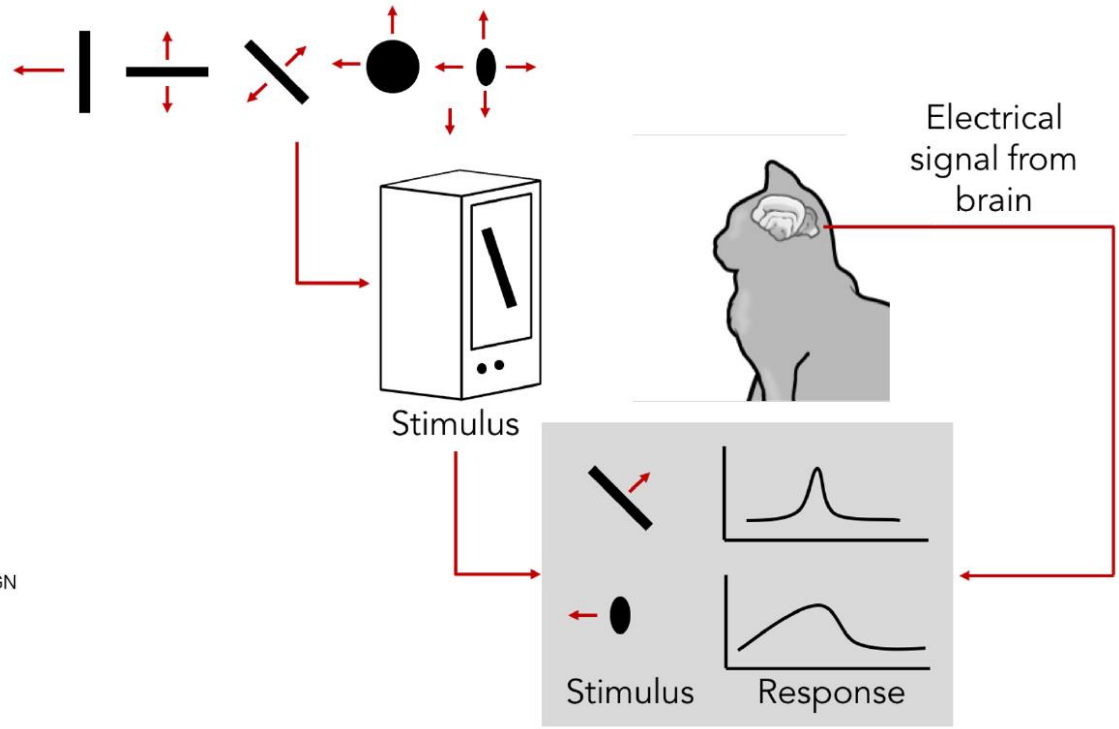
RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

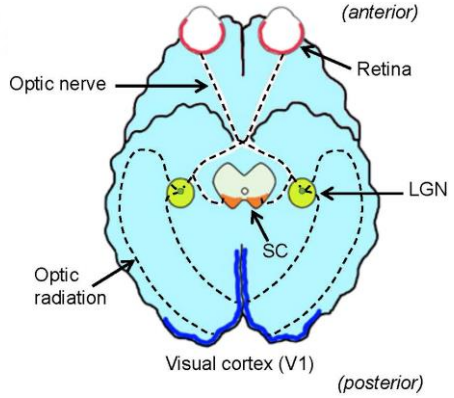
By D. H. HUBEL and T. N. WIESEL
From the Neurophysiology Laboratory, Department of Pharmacology Harvard Medical School, Boston, Massachusetts, U.S.A.

A BIT OF HISTORY

Hubel & Wiesel



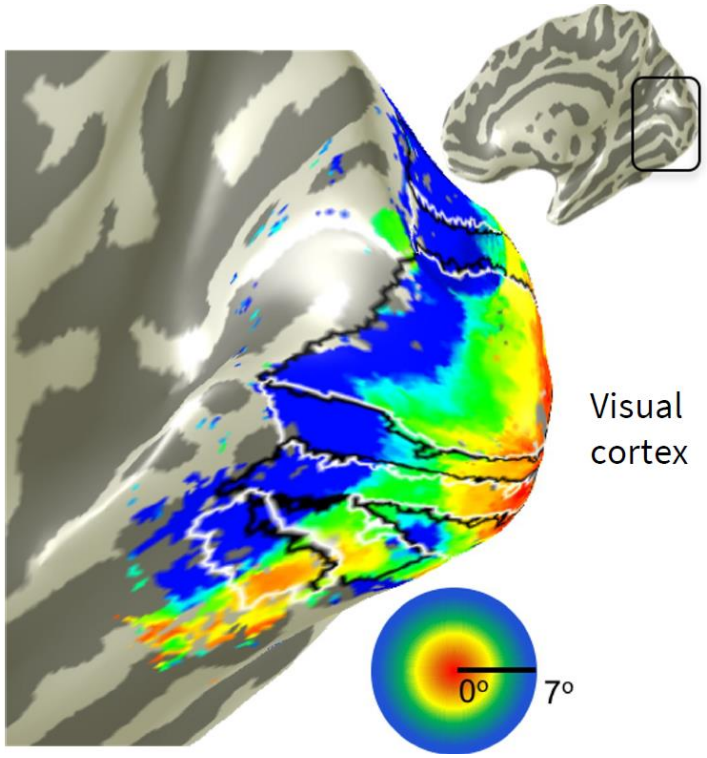




Slide inspiration: Fei-Fei Li, Ehsan Adeli

A BIT OF HISTORY

Topographical mapping in the cortex:
nearby cells in cortex represent
nearby regions in the visual field



Retinotopy images courtesy of Jesse Gomez in the Stanford Vision & Perception Neuroscience Lab.

Slide inspiration: Fei-Fei Li, Ehsan Adeli

A BIT OF HISTORY

Hierarchical organization

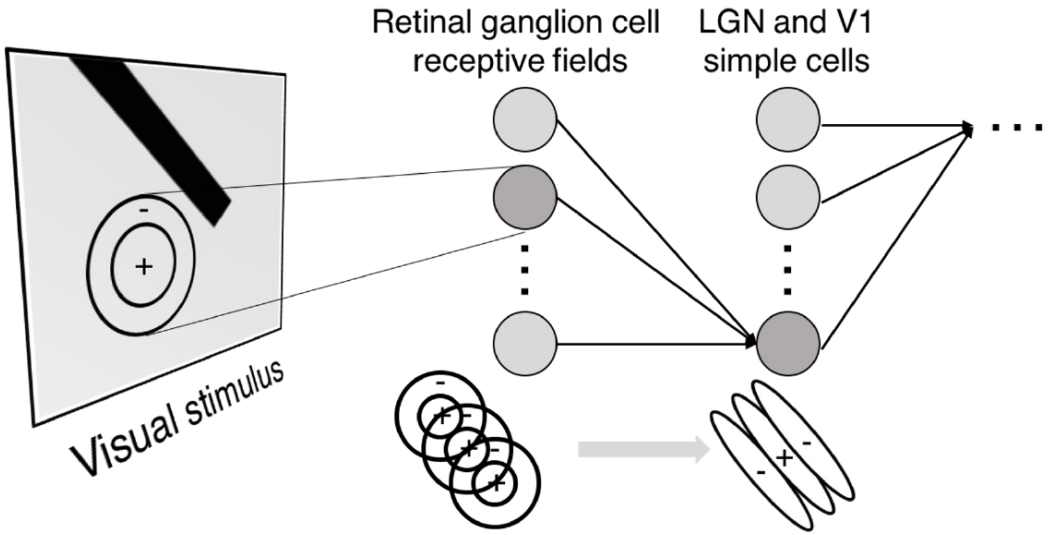


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point

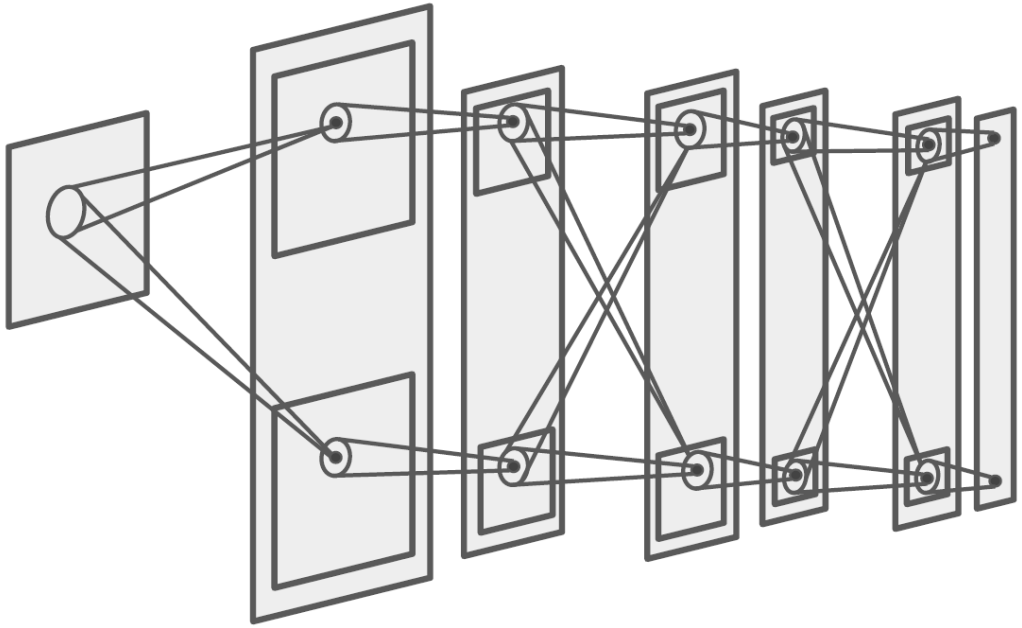
No response

Response (end point)

Slide inspiration: Fei-Fei Li, Ehsan Adeli

A BIT OF HISTORY

Neocognitron
[Fukushima 1980]



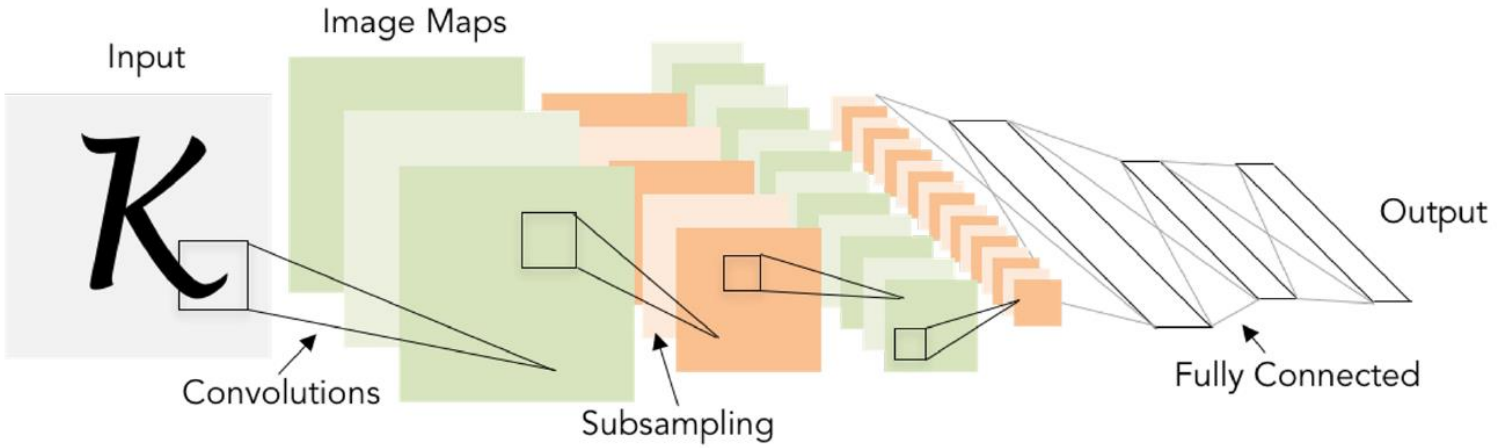
"sandwich" architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling

Slide inspiration: Fei-Fei Li, Ehsan Adeli

A BIT OF HISTORY

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]

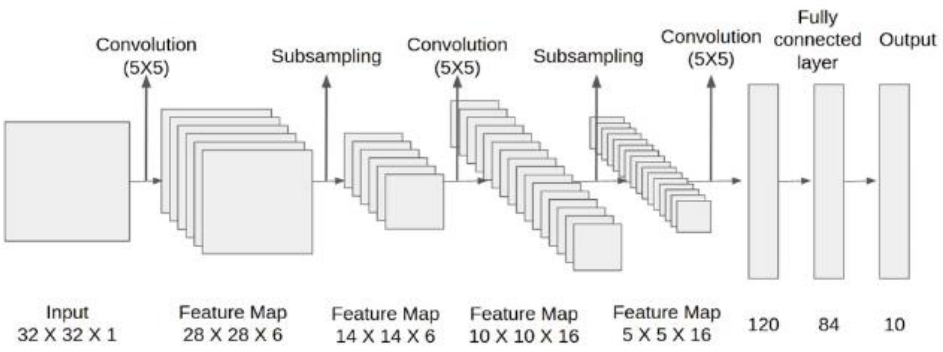


LeNet-5

Slide inspiration: Fei-Fei Li, Ehsan Adeli

A BIT OF HISTORY

- LeNet-5 (1998)
  - 7 layers
  - Input: 32×32 pixel image Gray level (one channel)
  - 60,000 trainable free parameters (weight sharing)
  - Mnist dataset→ 60,000 training examples , 10,000 test examples
  - Training time: 2 to 3 days with CPU



Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

A BIT OF HISTORY

ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]

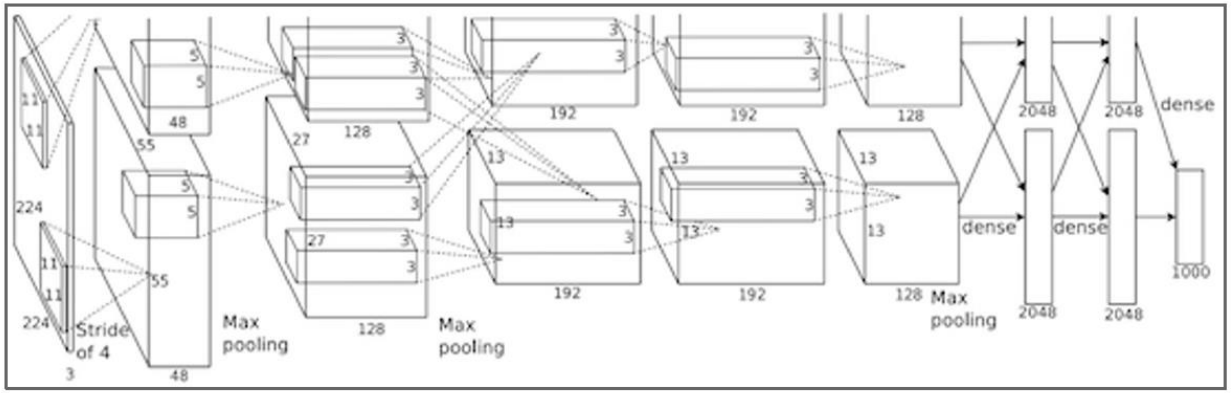


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

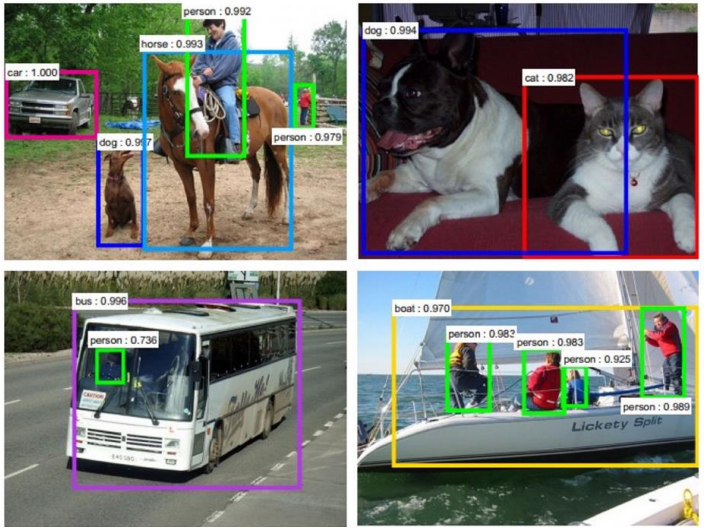
Slide inspiration: Fei-Fei Li, Ehsan Adeli

FAST-FORWARD: CONVNETS ARE EVERYWHERE

Classification

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

Detection



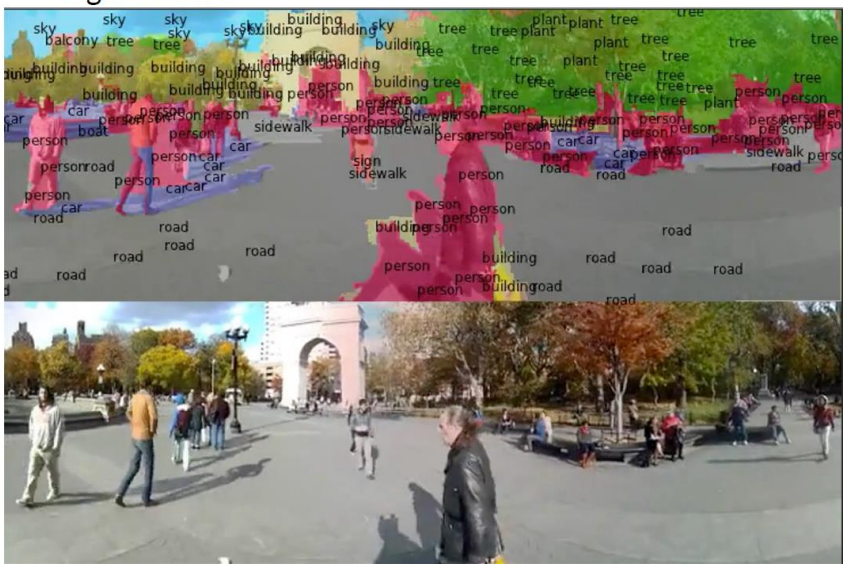
Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Slide inspiration: Fei-Fei Li, Ehsan Adeli

FAST-FORWARD: CONVNETS ARE EVERYWHERE

Segmentation



Figures copyright Clement Farabet, 2012.
Reproduced with permission.
[Farabet et al., 2012]

self-driving cars

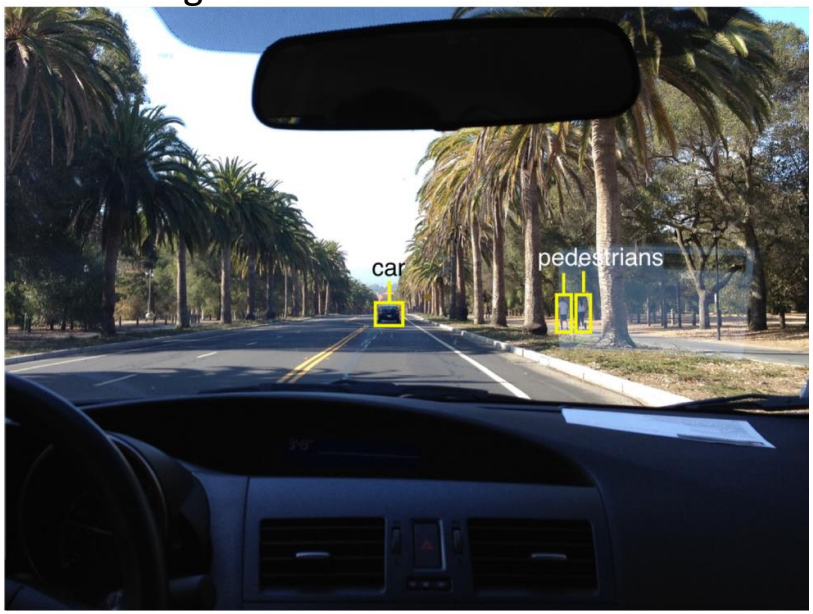


Photo by Lane McIntosh. Copyright CS231n 2017.

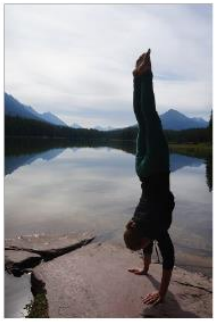
Slide inspiration: Fei-Fei Li, Ehsan Adeli

FAST-FORWARD: CONVNETS ARE EVERYWHERE

Image Captioning

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor



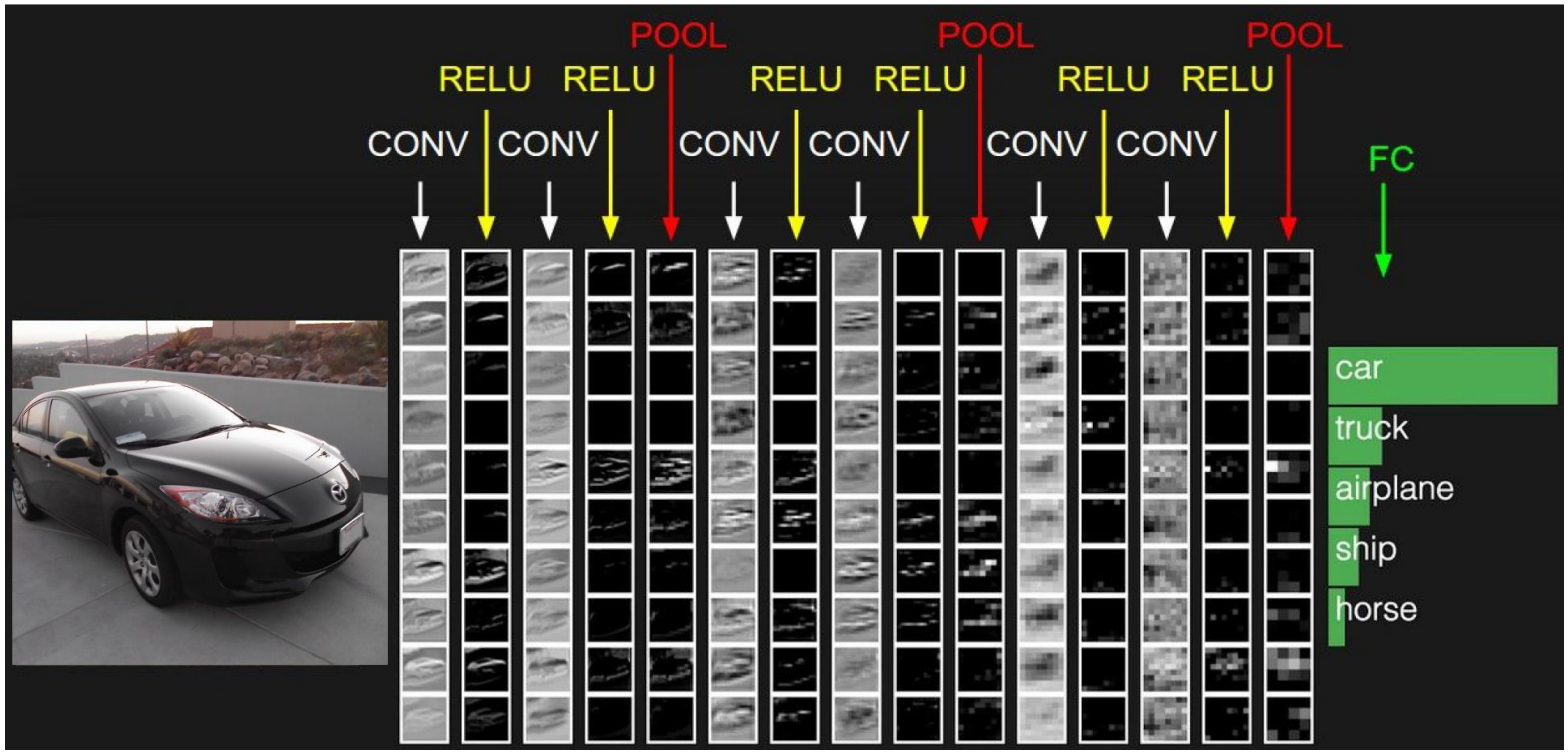
A woman standing on a beach holding a surfboard

[Vinyals et al., 2015]
[Karpathy and Fei-Fei,2015]

Slide inspiration: Fei-Fei Li, Ehsan Adeli

Convolutional Neural Networks (CNNs)

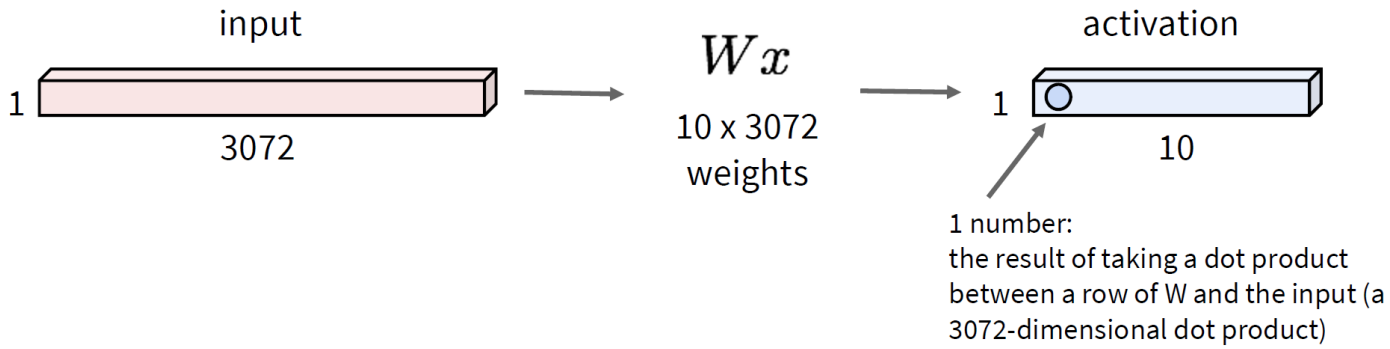
CONVOLUTIONAL NEURAL NETWORKS (CNNS)



Slide inspiration: Fei-Fei Li, Ehsan Adeli

FULLY CONNECTED LAYER

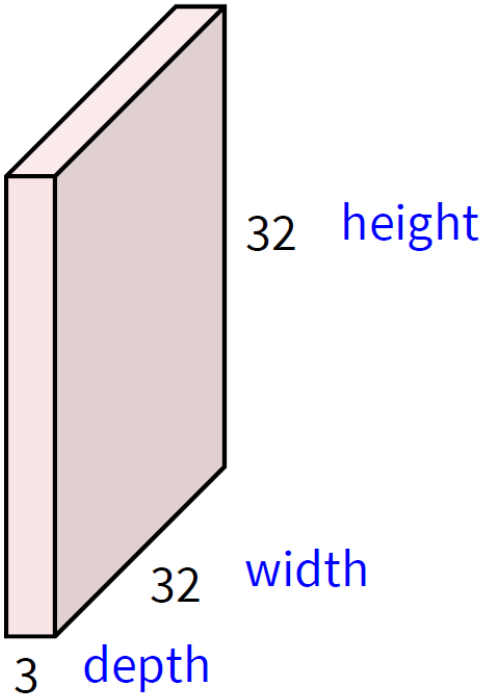
32x32x3 image -> stretch to 3072 x 1



Slide inspiration: Fei-Fei Li, Ehsan Adeli

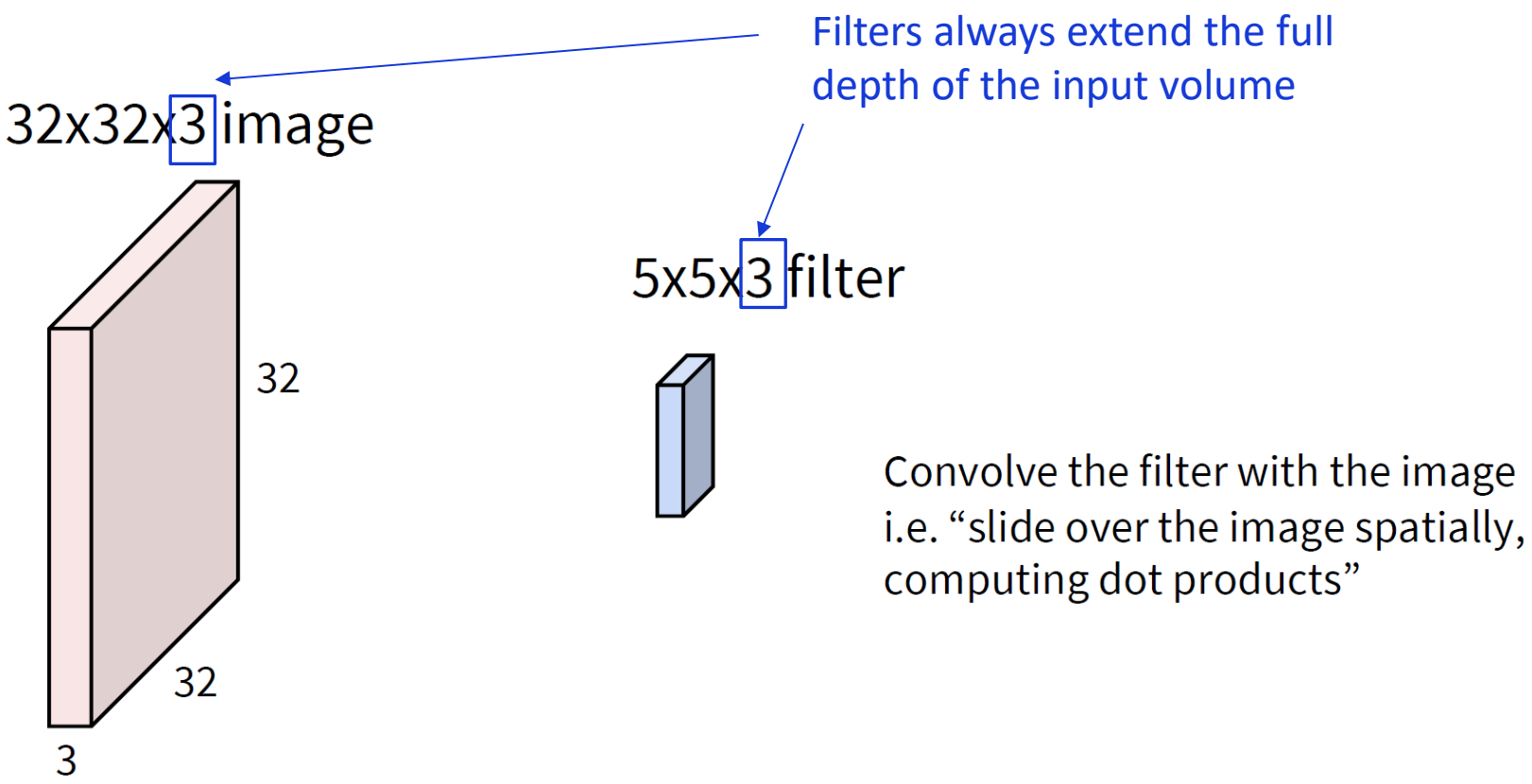
CONVOLUTION LAYER

32x32x3 image -> preserve spatial structure



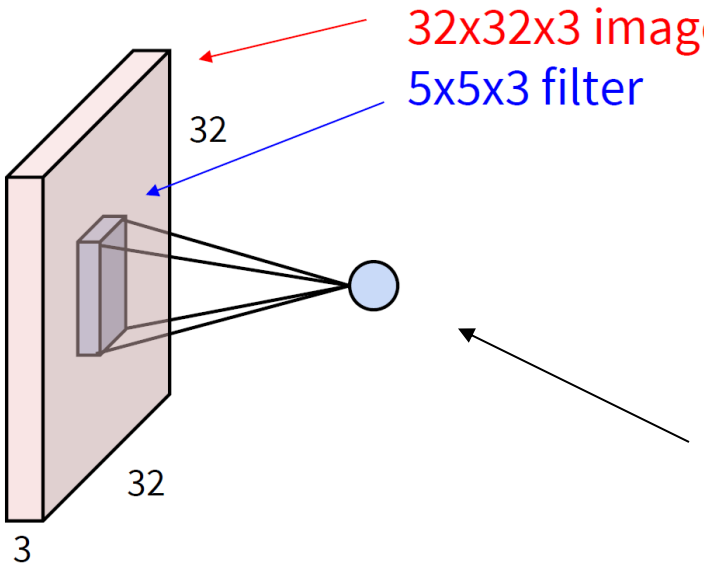
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

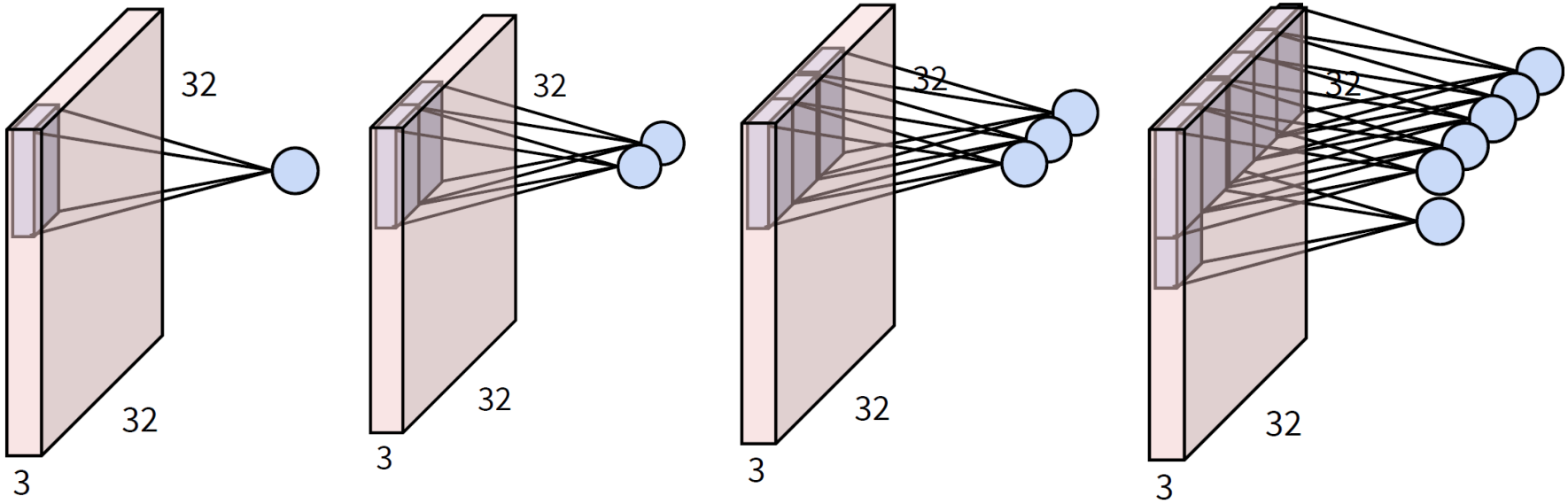


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

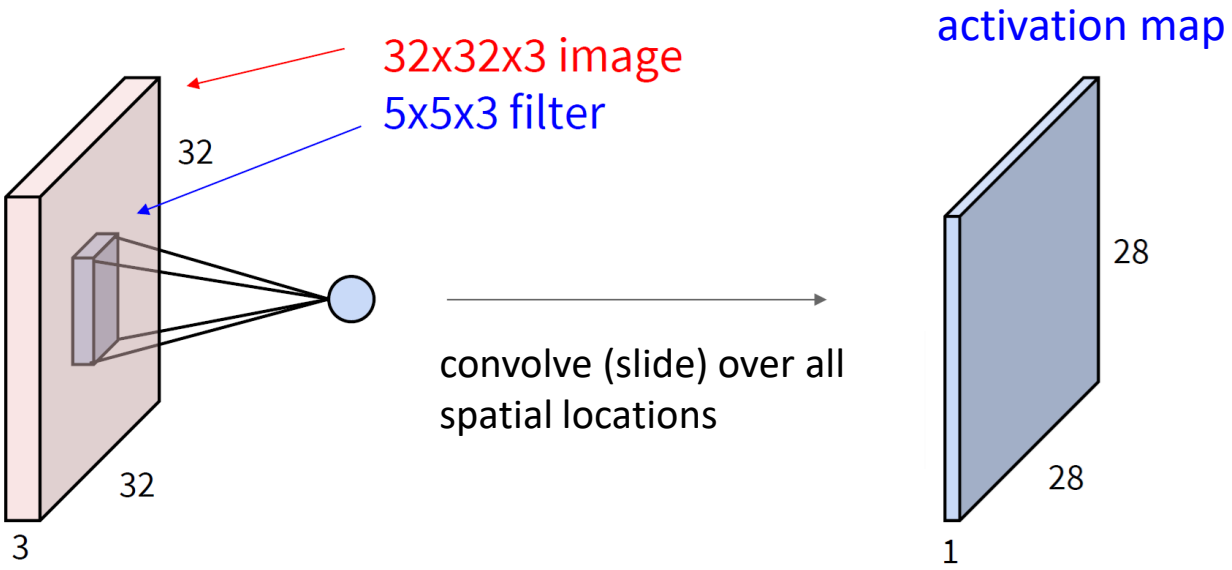
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



Slide inspiration: Fei-Fei Li, Ehsan Adeli

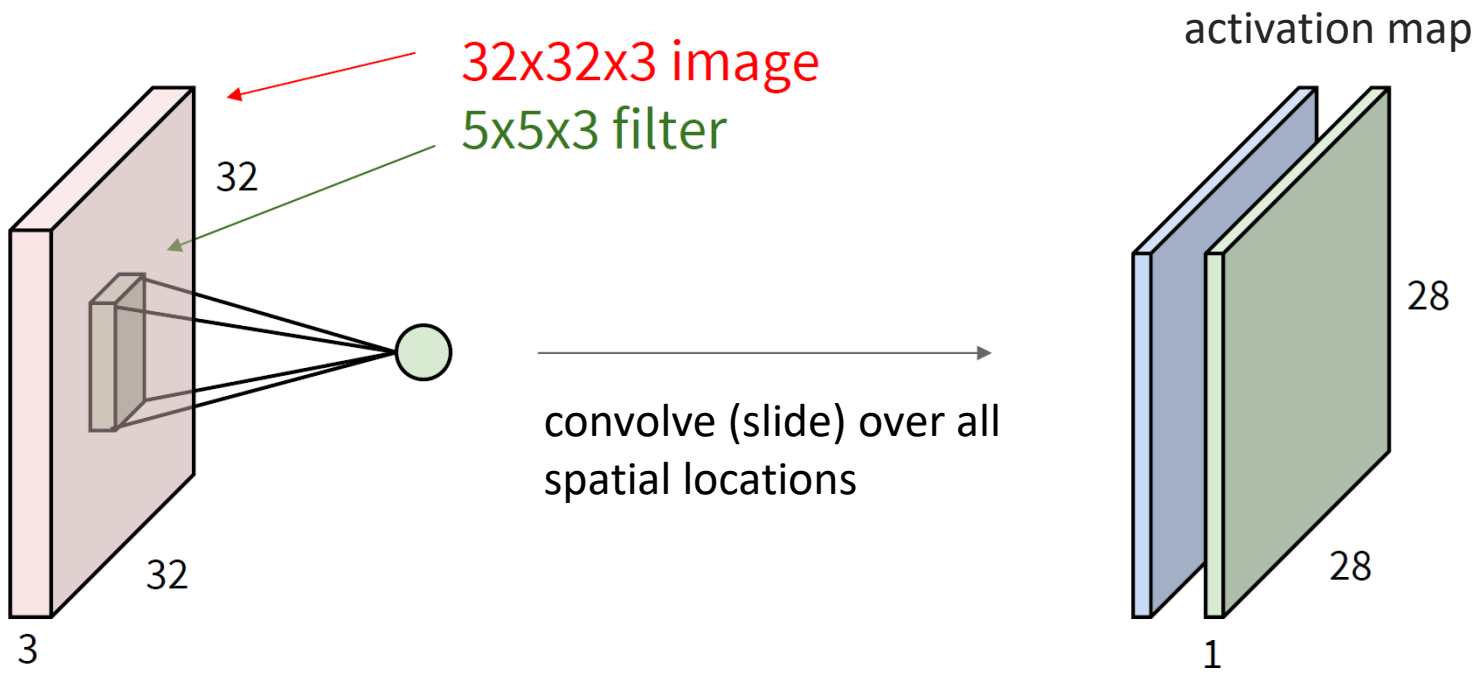
CONVOLUTION LAYER



Slide inspiration: Fei-Fei Li, Ehsan Adeli

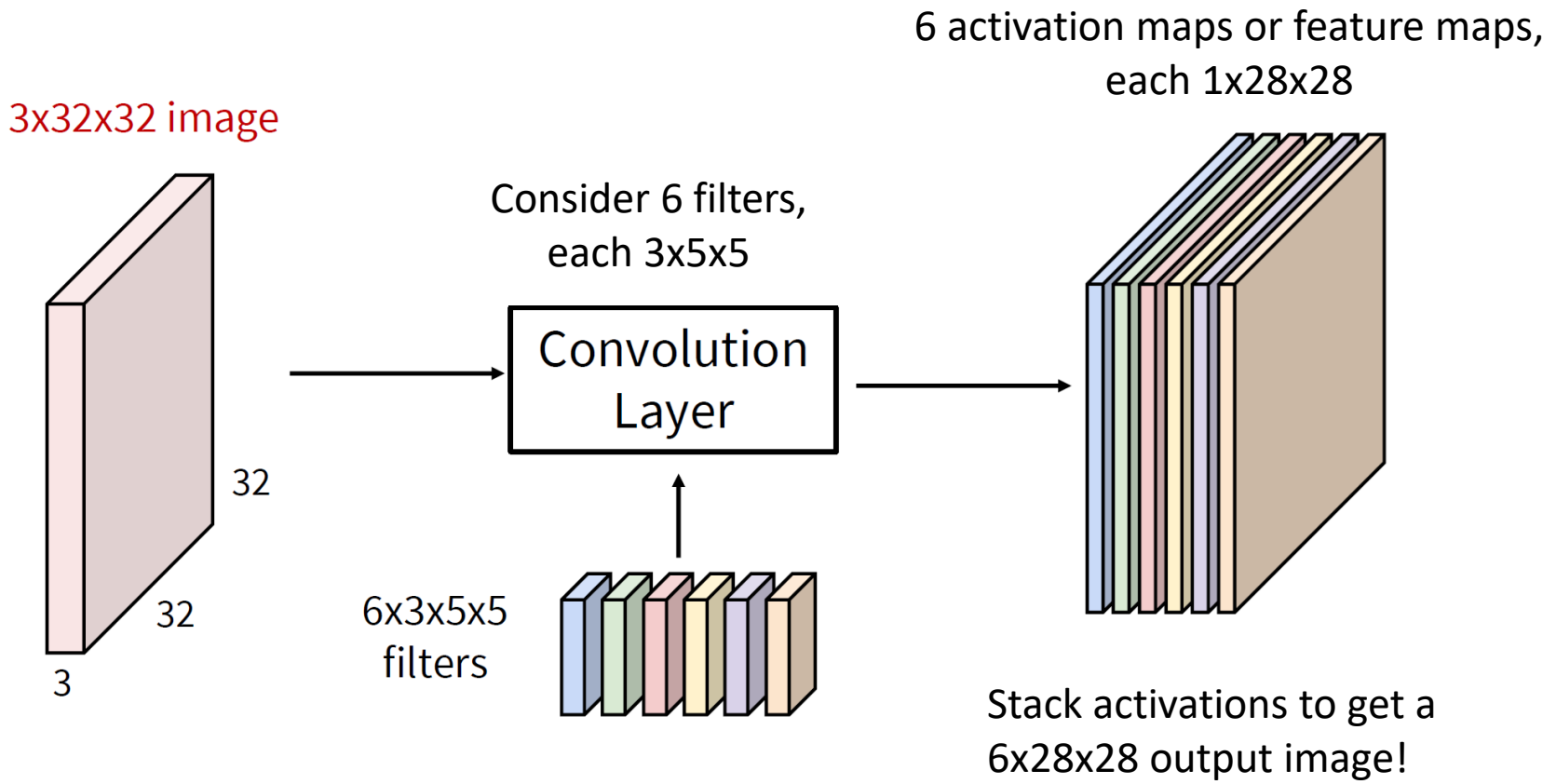
CONVOLUTION LAYER

consider a second, green filter



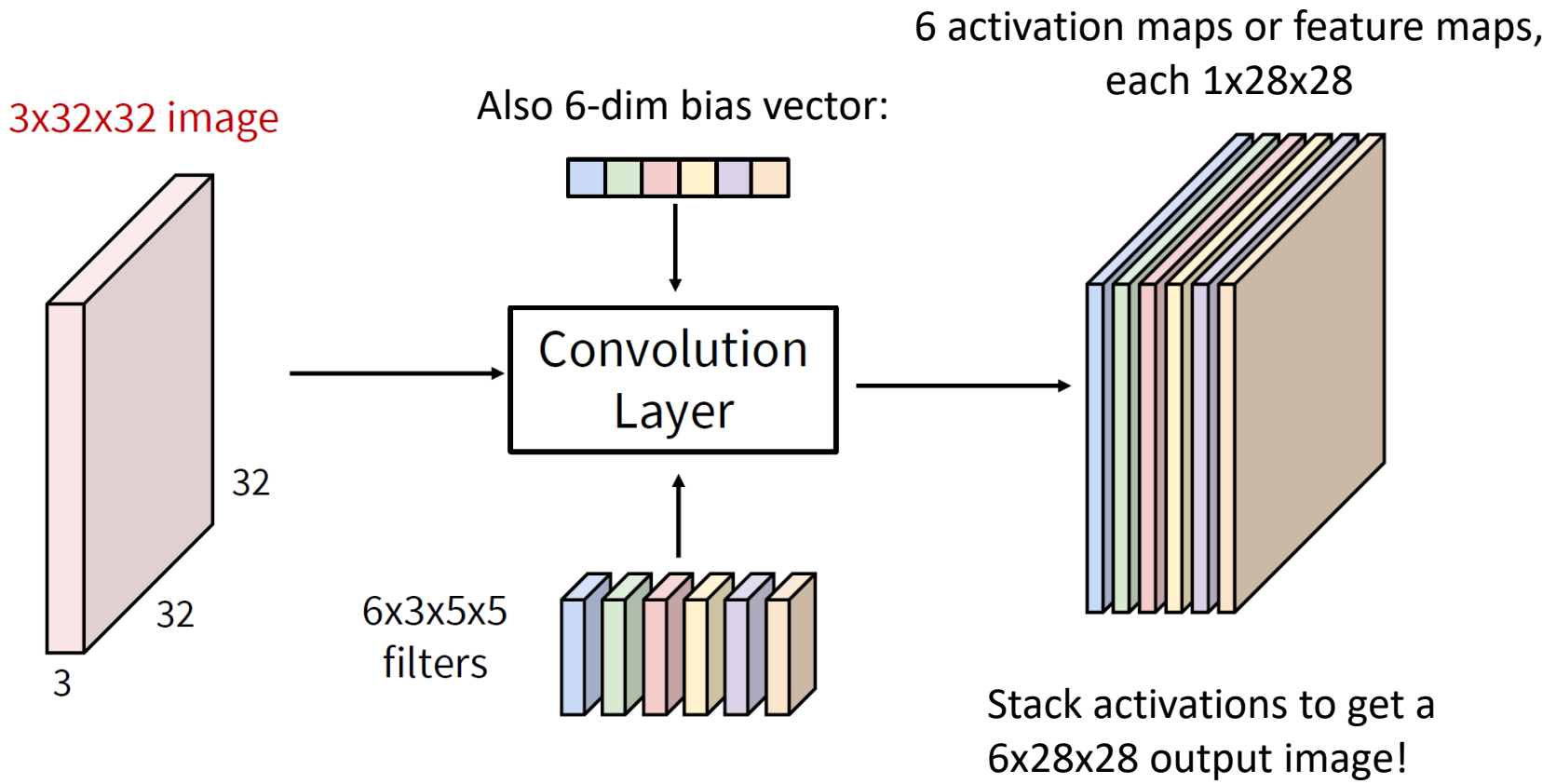
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



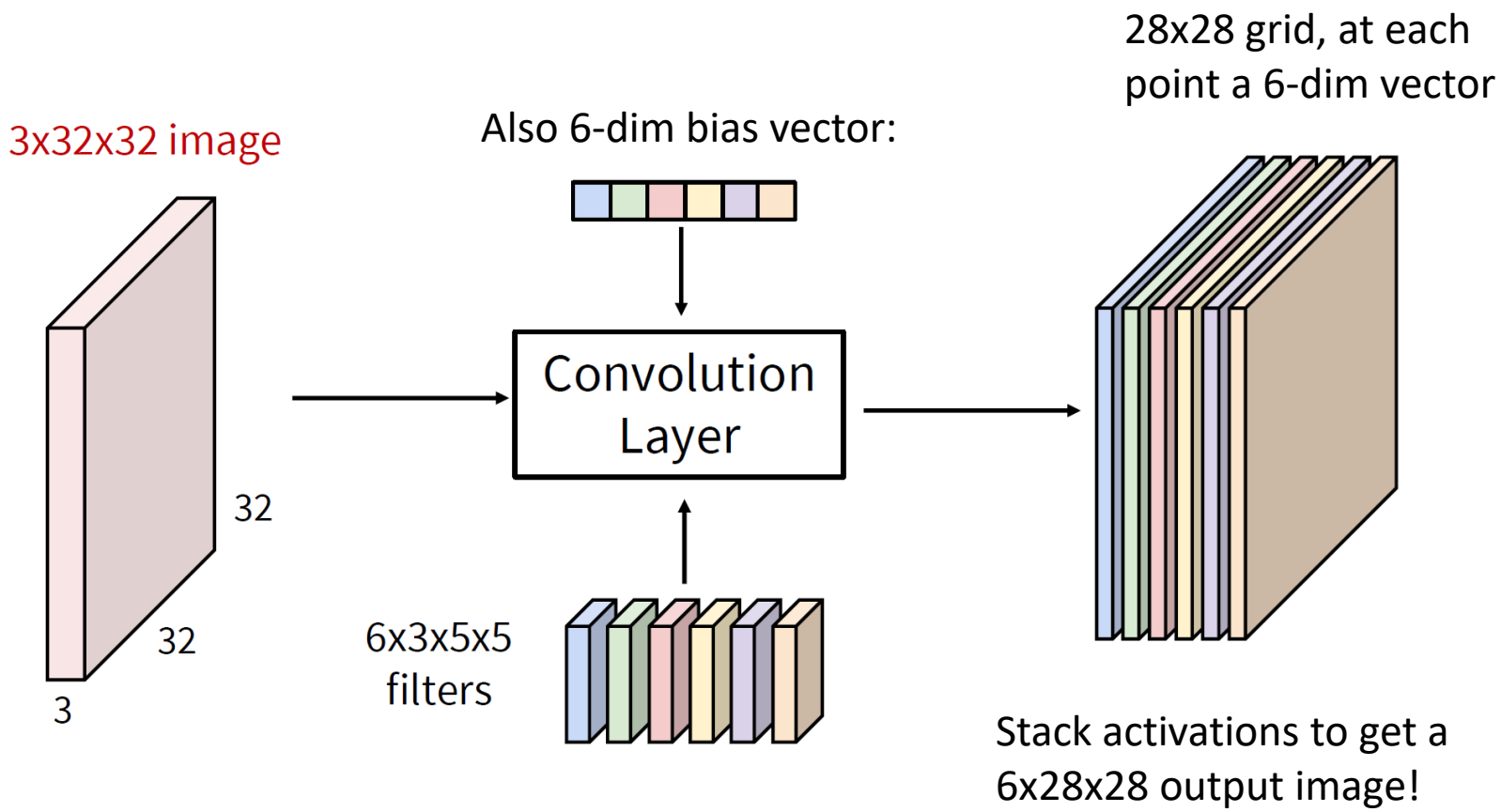
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



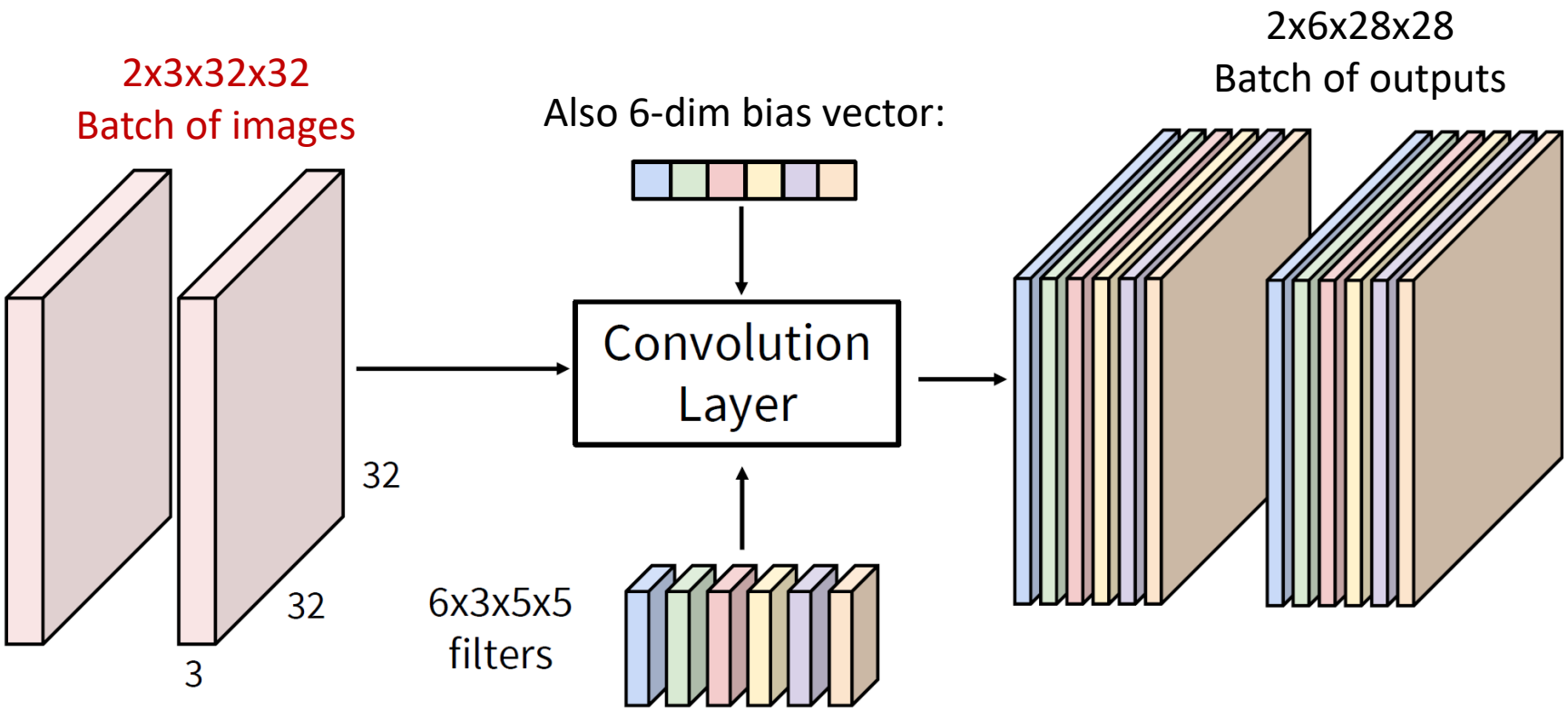
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



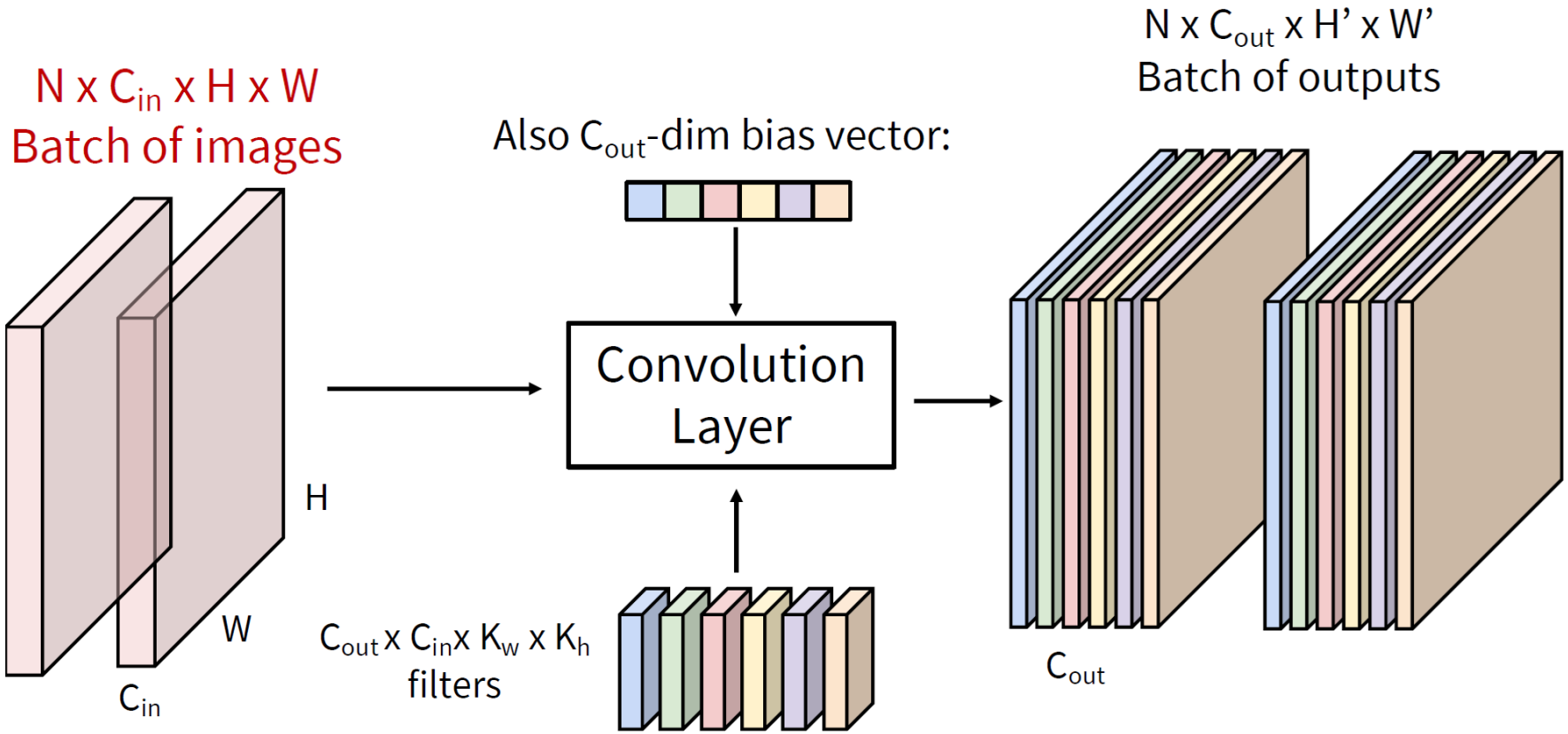
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



Slide inspiration: Fei-Fei Li, Ehsan Adeli

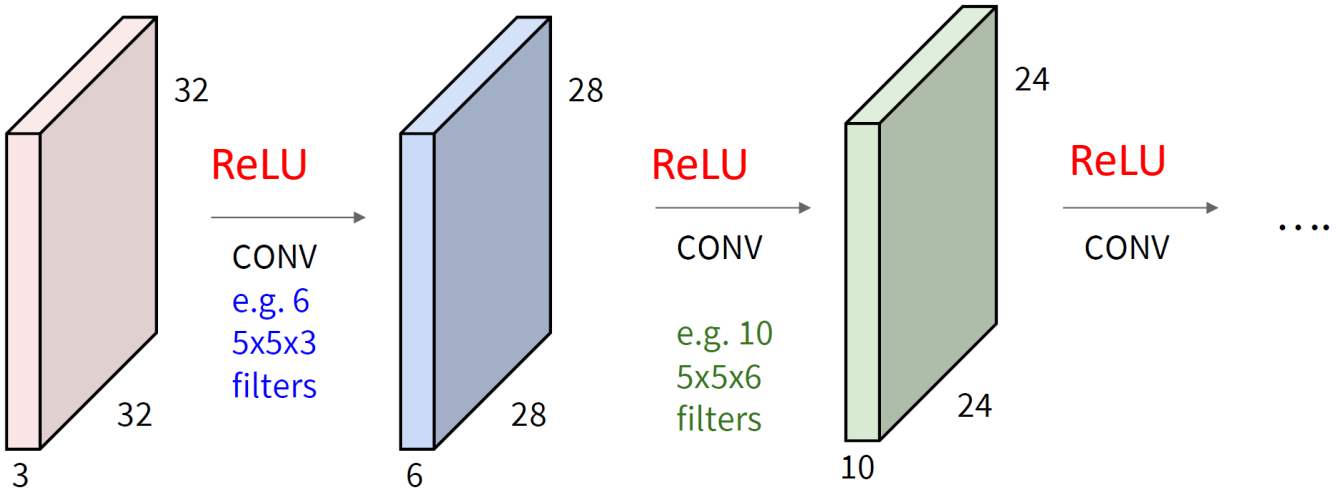
CONVOLUTION LAYER



Slide inspiration: Fei-Fei Li, Ehsan Adeli

PREVIEW: CONVNET IS A SEQUENCE OF CONVOLUTION LAYERS

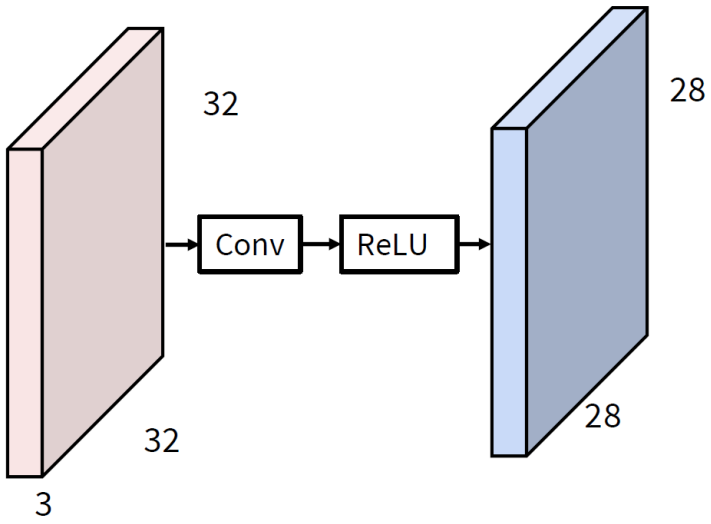
interspersed with activation functions



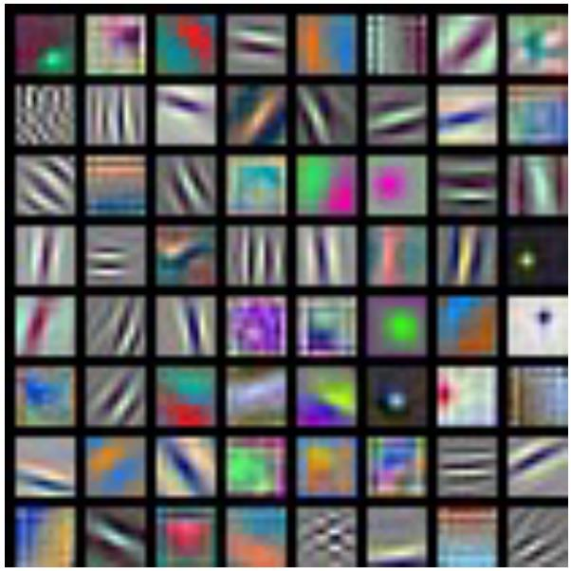
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Preview: What do convolutional filters learn?



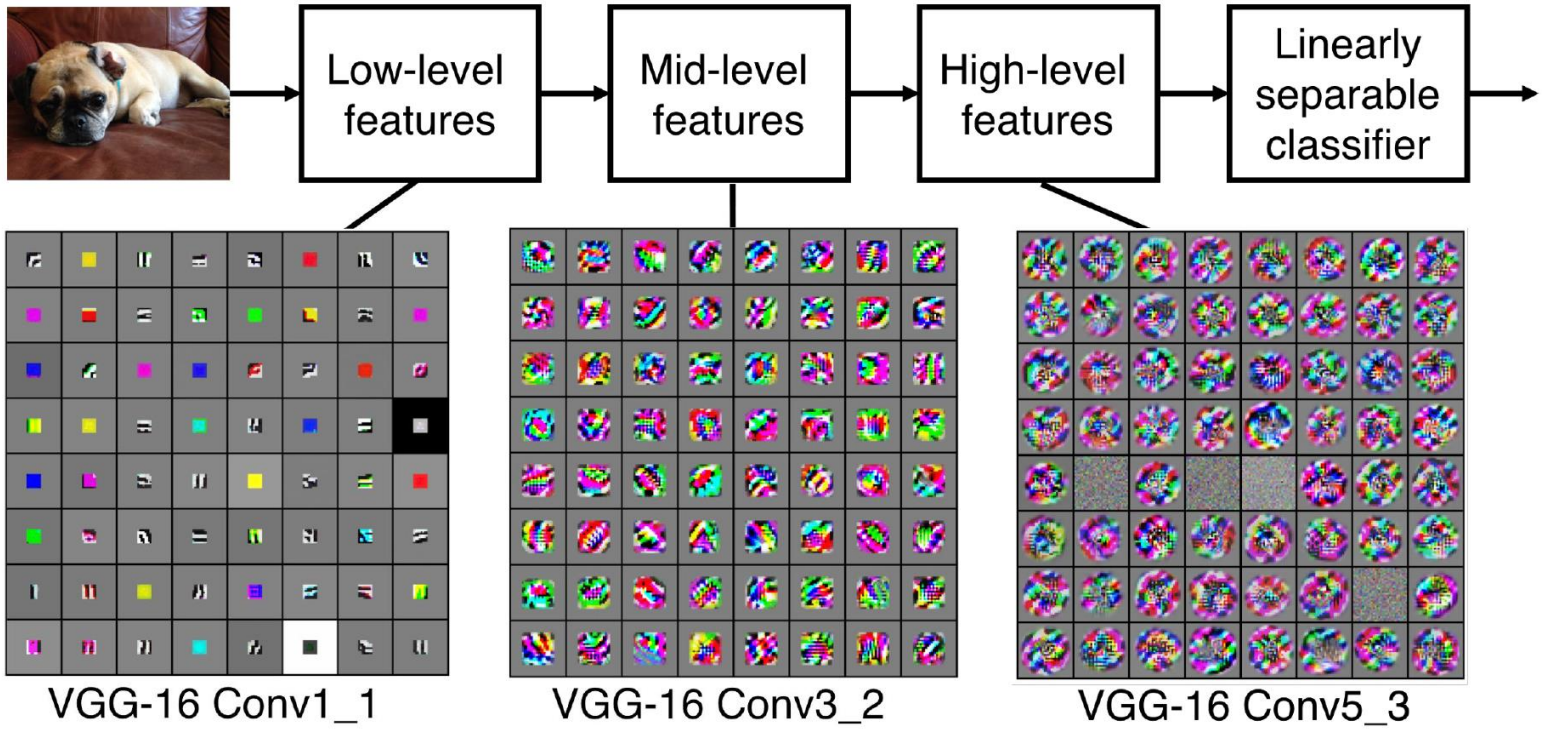
First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)



AlexNet: 64 filters, each 3x11x11

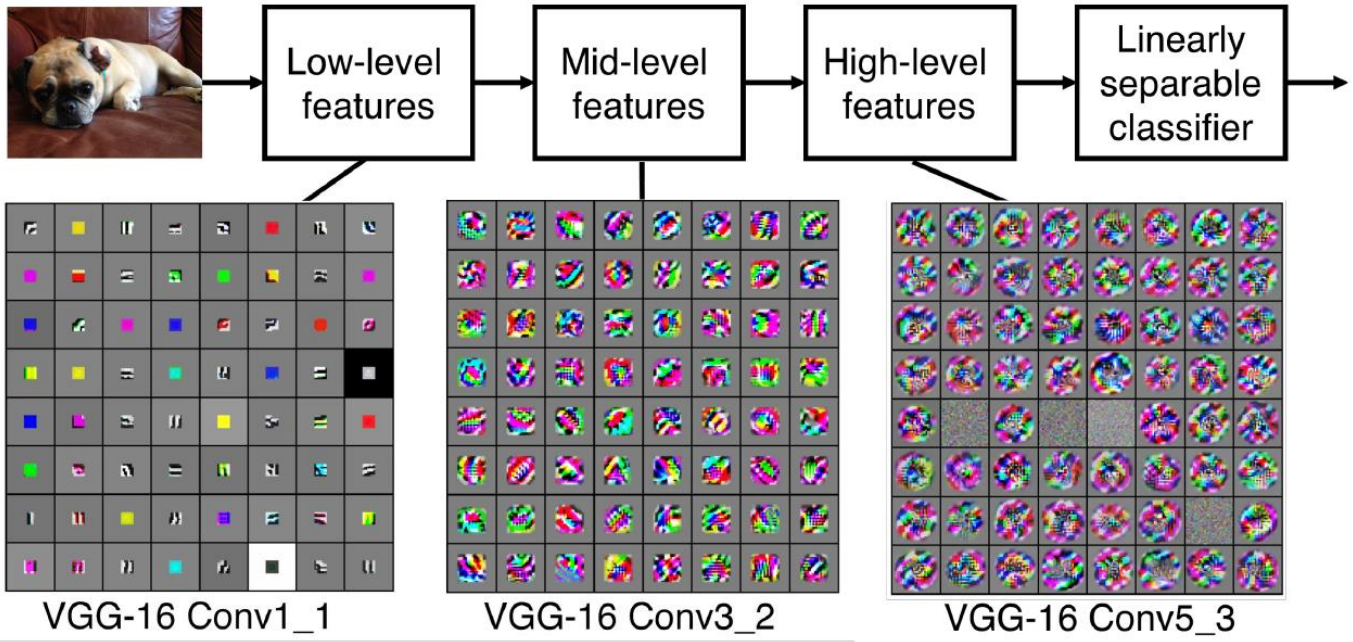
Slide inspiration: Fei-Fei Li, Ehsan Adeli

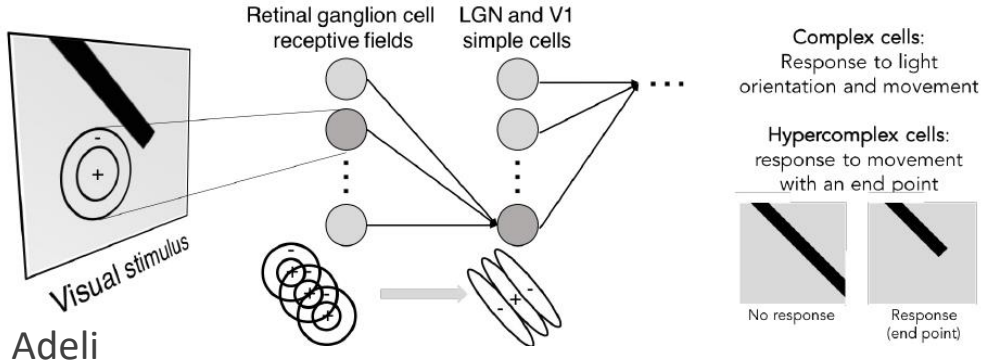
CONVOLUTION LAYER



Slide inspiration: Fei-Fei Li, Ehsan Adeli

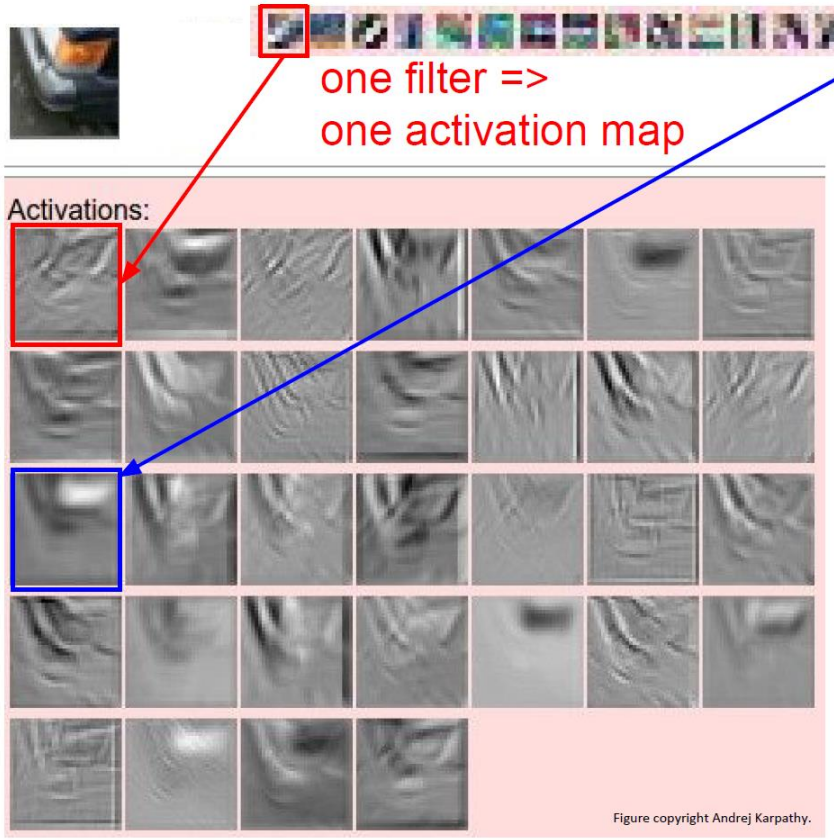
CONVOLUTION LAYER





Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



example 5x5 filters
(32 total)

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

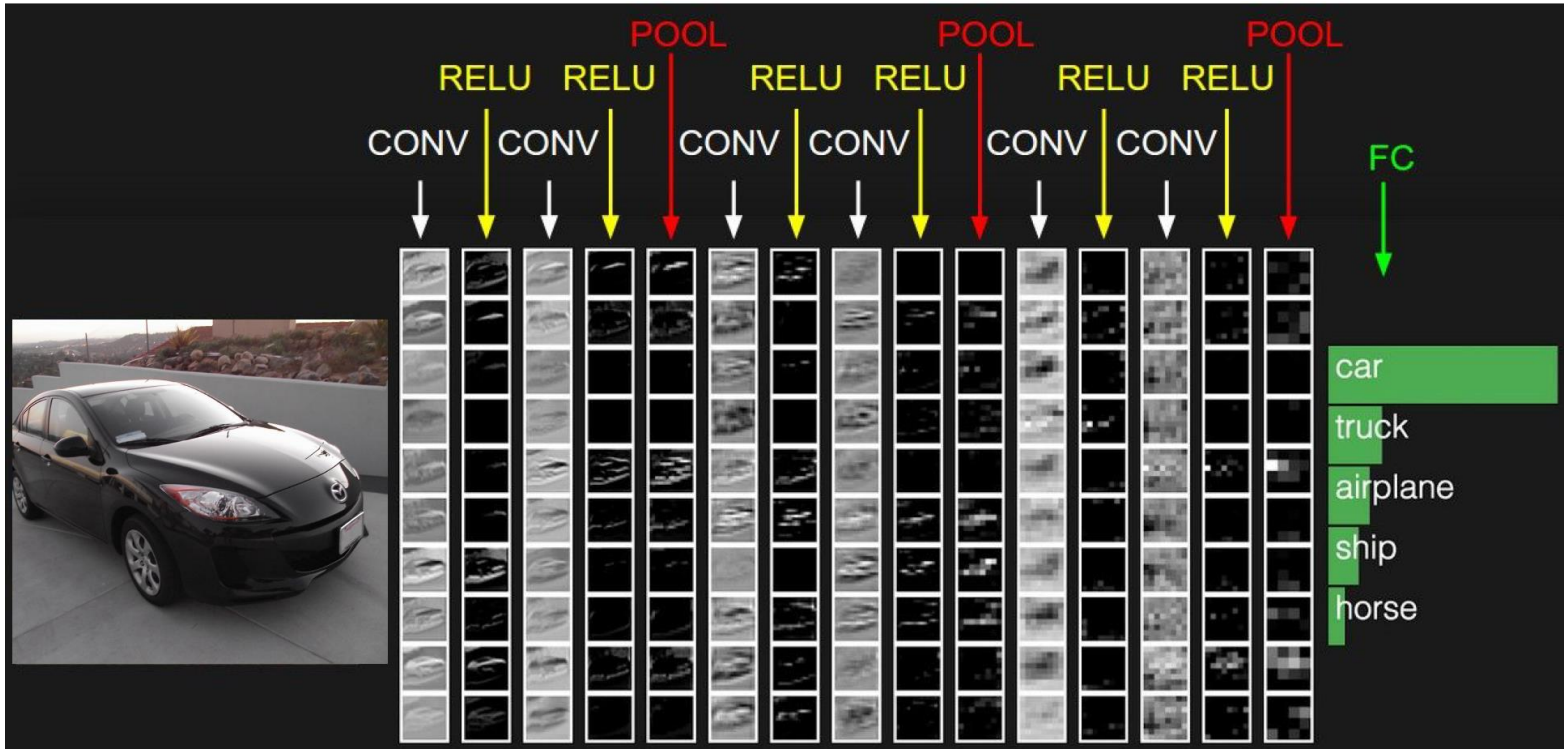
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

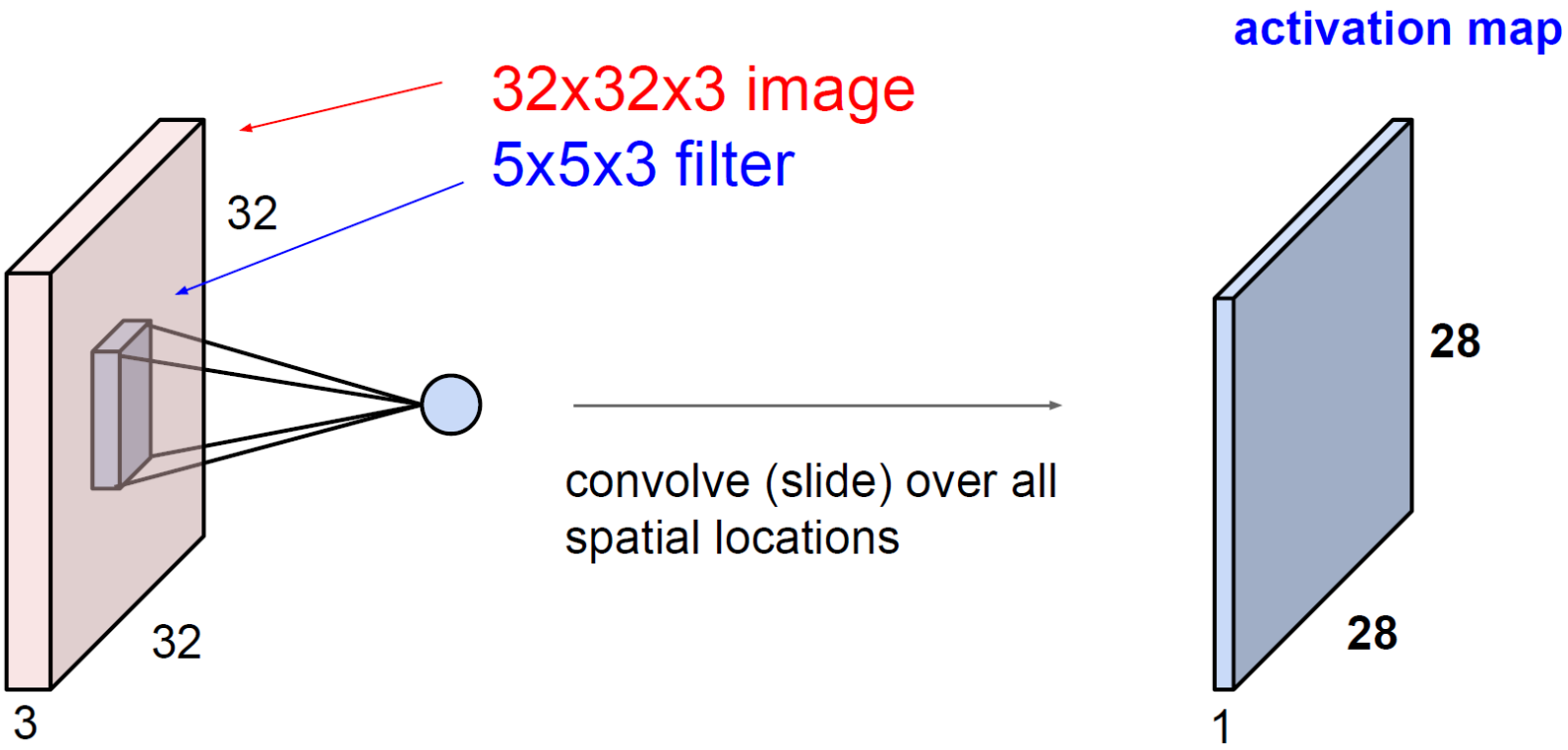
- **Preview:**



Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:

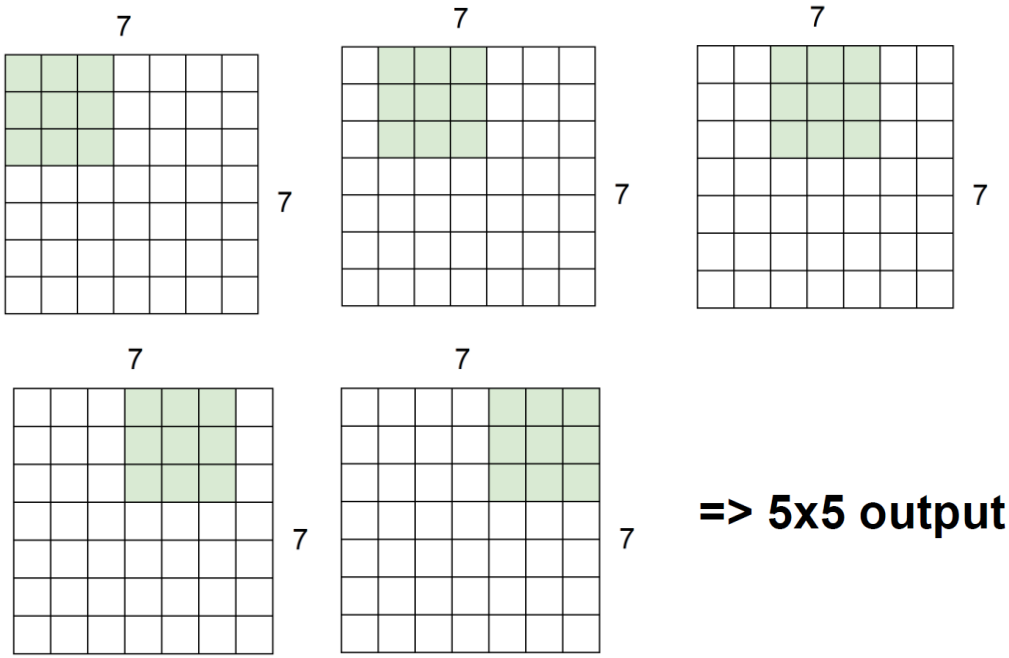


Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter



=> 5x5 output

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

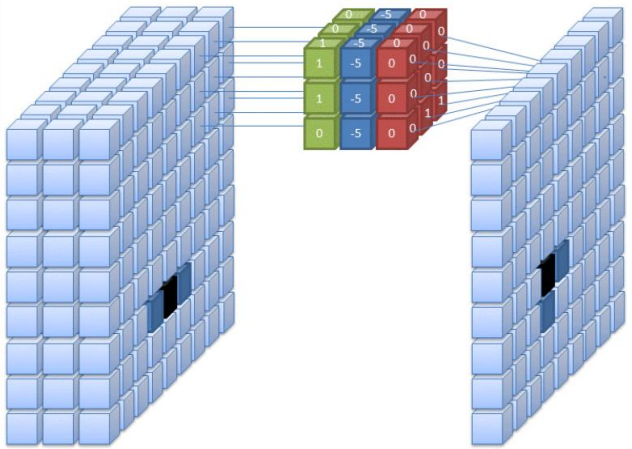
o[:,:,0]

2	3	3
3	7	3
8	10	-3

o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

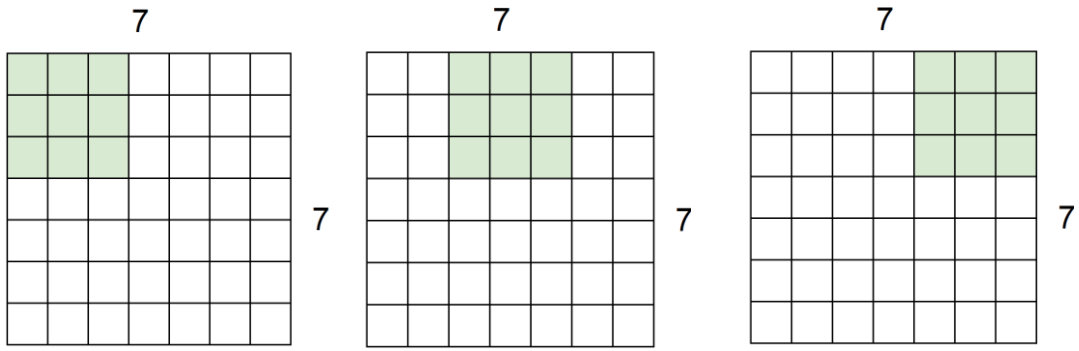
toggle movement



Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:

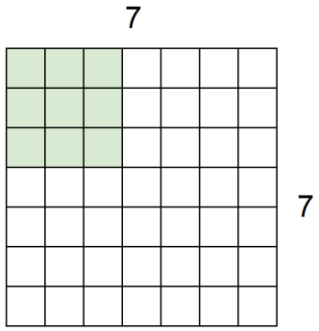


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:



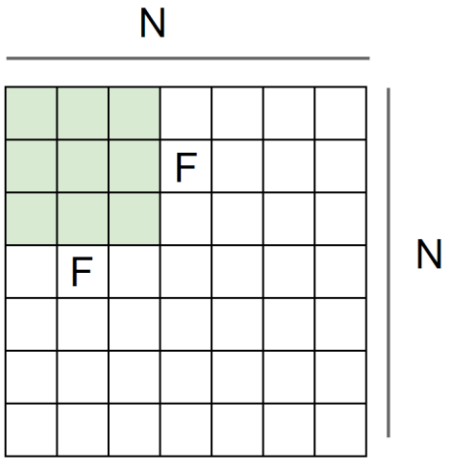
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:



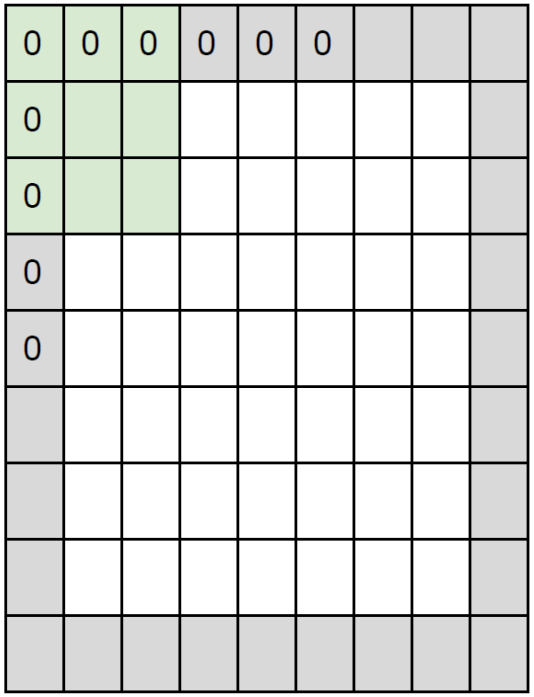
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:



In practice: Common to zero pad the border

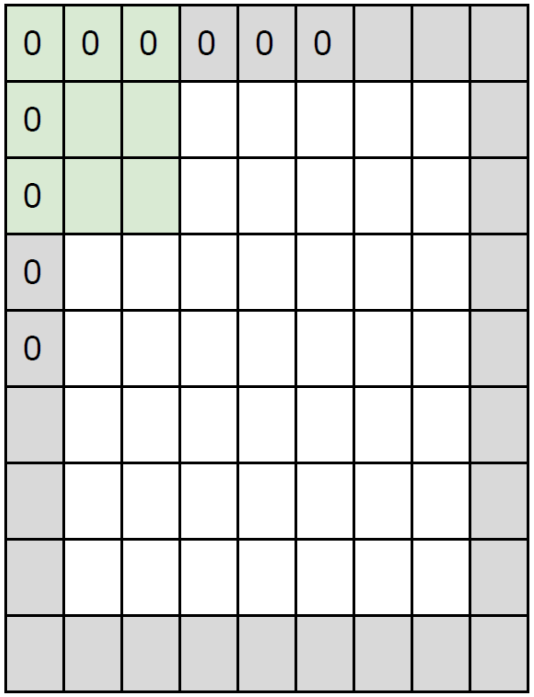
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:

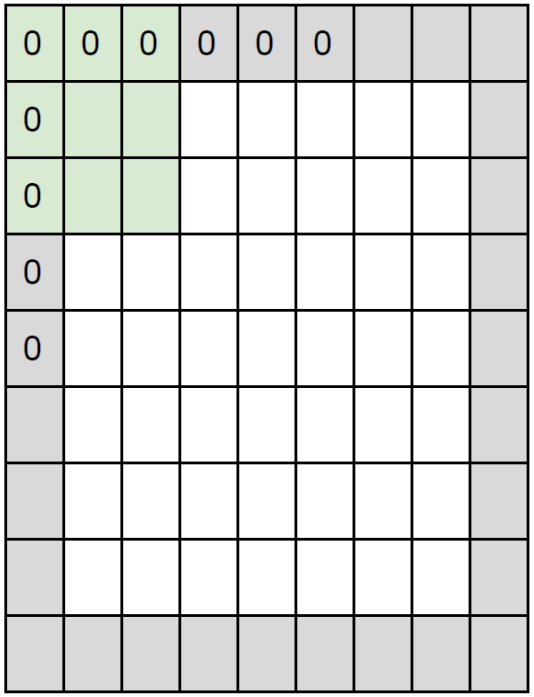


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?
7x7 output!

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

A closer look at spatial dimensions:



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?
7x7 output!

in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

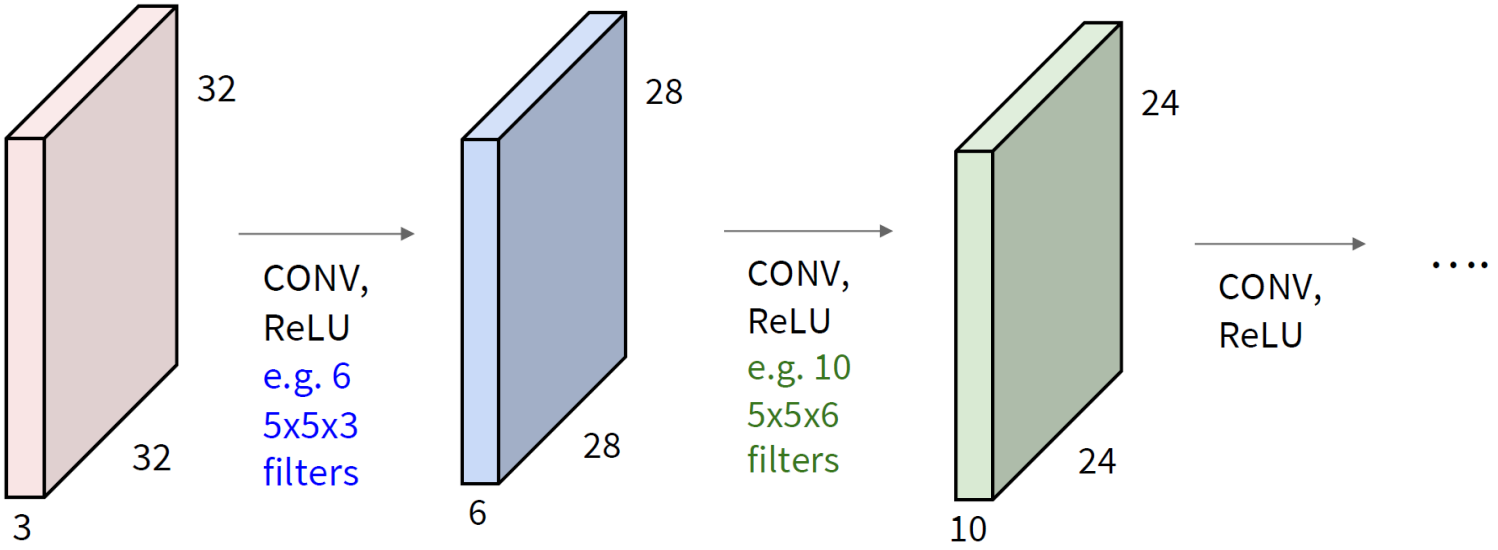
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Remember back to…

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially!

(32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



Slide inspiration: Fei-Fei Li, Ehsan Adeli

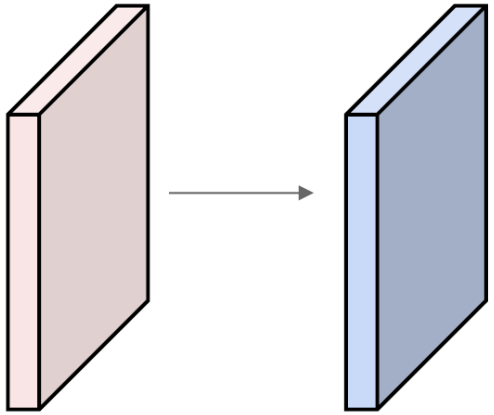
CONVOLUTION LAYER

Examples:

Input volume: 32x32x3

10 5x5 filters with stride 1, pad 2

Output volume size: ?



Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Examples:

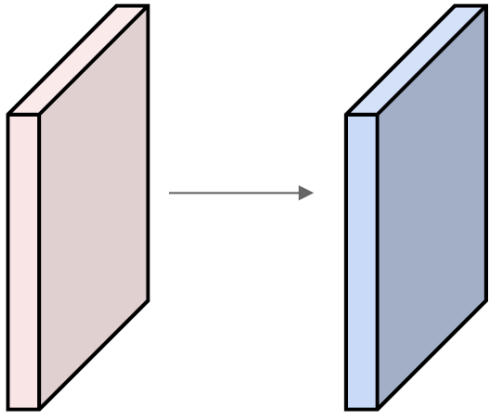
Input volume: 32x32x3

10 5x5 filters with stride 1, pad 2

Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so

32x32x10



Slide inspiration: Fei-Fei Li, Ehsan Adeli

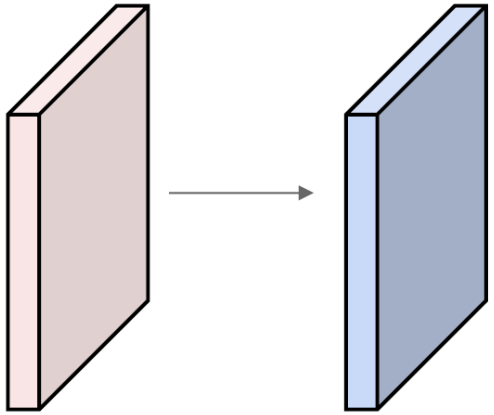
CONVOLUTION LAYER

Examples time:

Input volume: 32x32x3

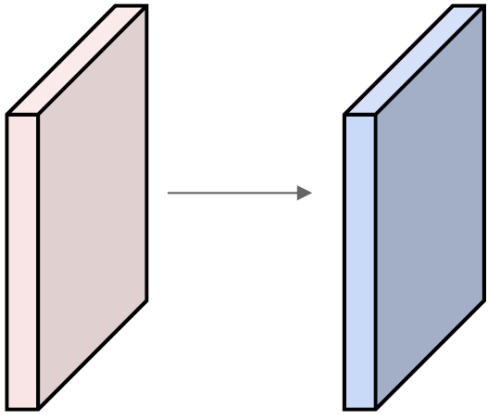
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER



Examples:

Input volume: 32x32x3

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

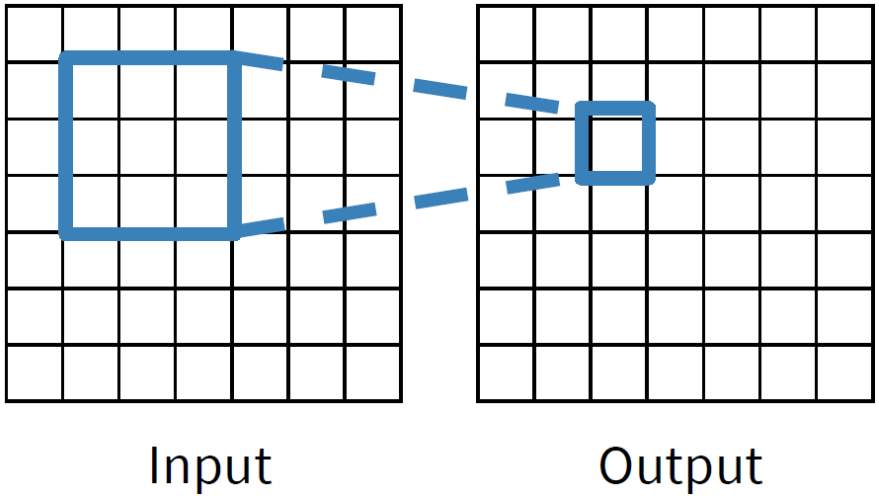
each filter has 5*5*3 + 1 = 76 params (+1 for bias) => 76*10 = 760

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Receptive Fields

For convolution with **kernel size K**, each element in the output depends on a K x K receptive field in the input

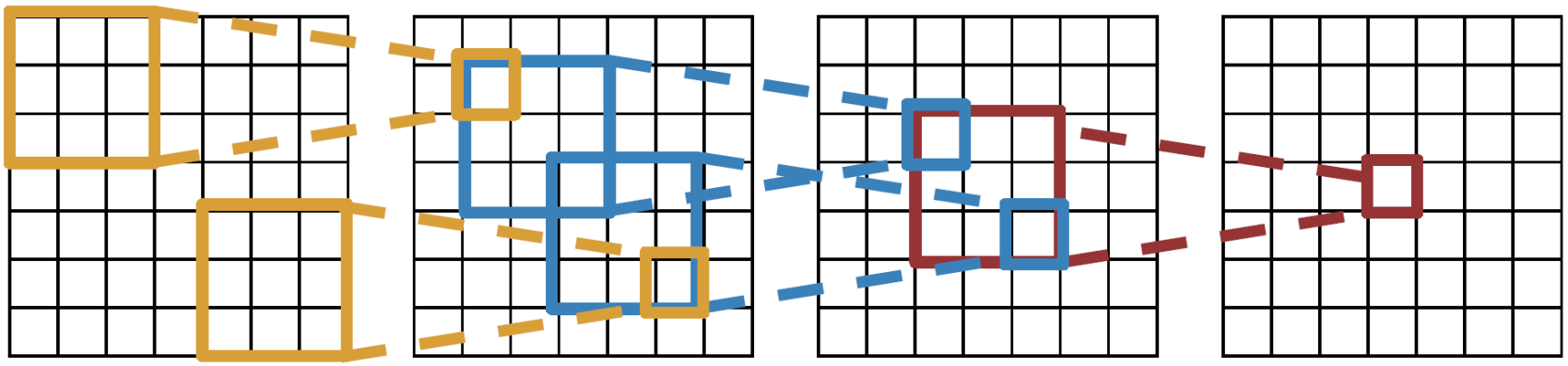


Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size With L layers the receptive field size is 1 + L * (K – 1)



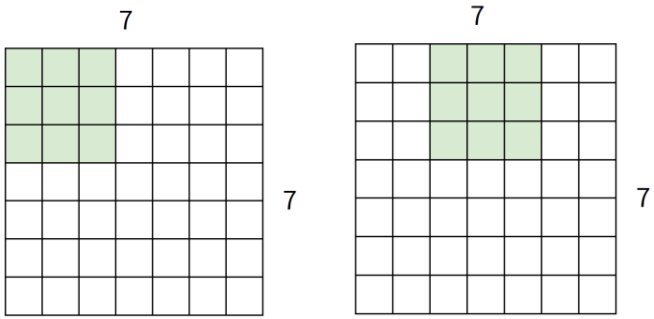
Problem: For large images we need many layers for each output to “see” the whole image

Solution: Downsample inside the network

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Solution: Strided Convolution



7x7 input (spatially)
assume 3x3 filter
applied with stride 2

=> 3x3 output!

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Convolution layer: summary

Let's assume input is W_1 x H_1 x C
Conv layer needs 4 hyperparameters:
- Number of filters K
- The filter size F
- The stride S
- The zero padding P

This will produce an output of W_2 x H_2 x K
where:
- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$

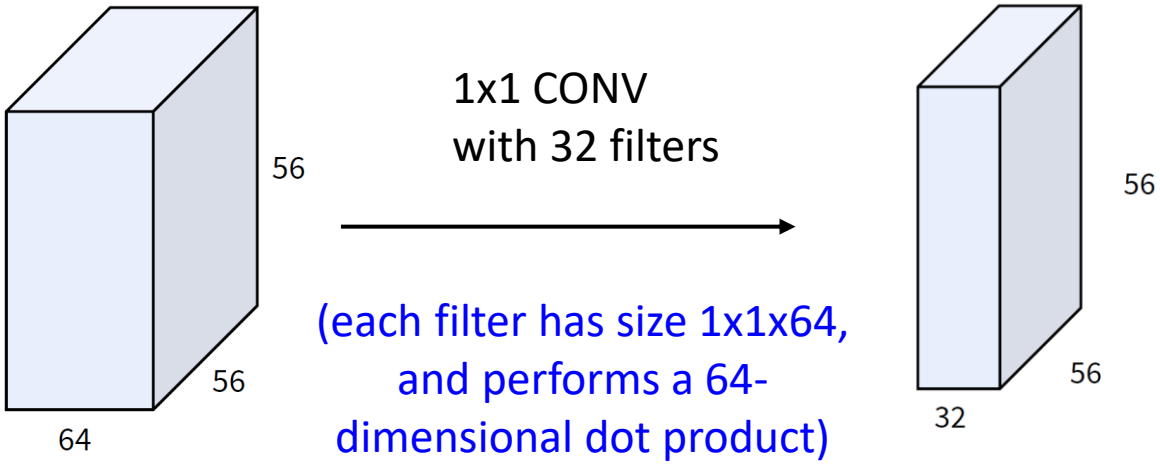
Number of parameters: F^2CK and K biases

Common settings:
K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

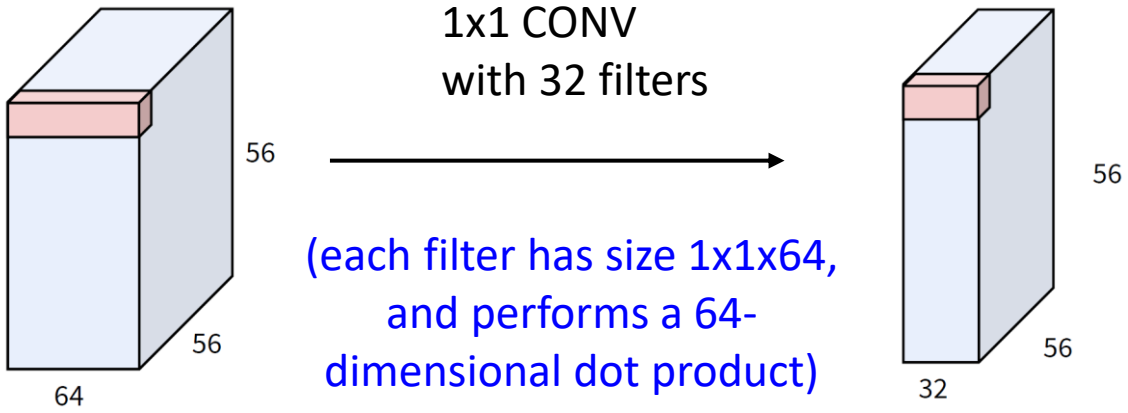
(1x1 convolution layers make perfect sense)



Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

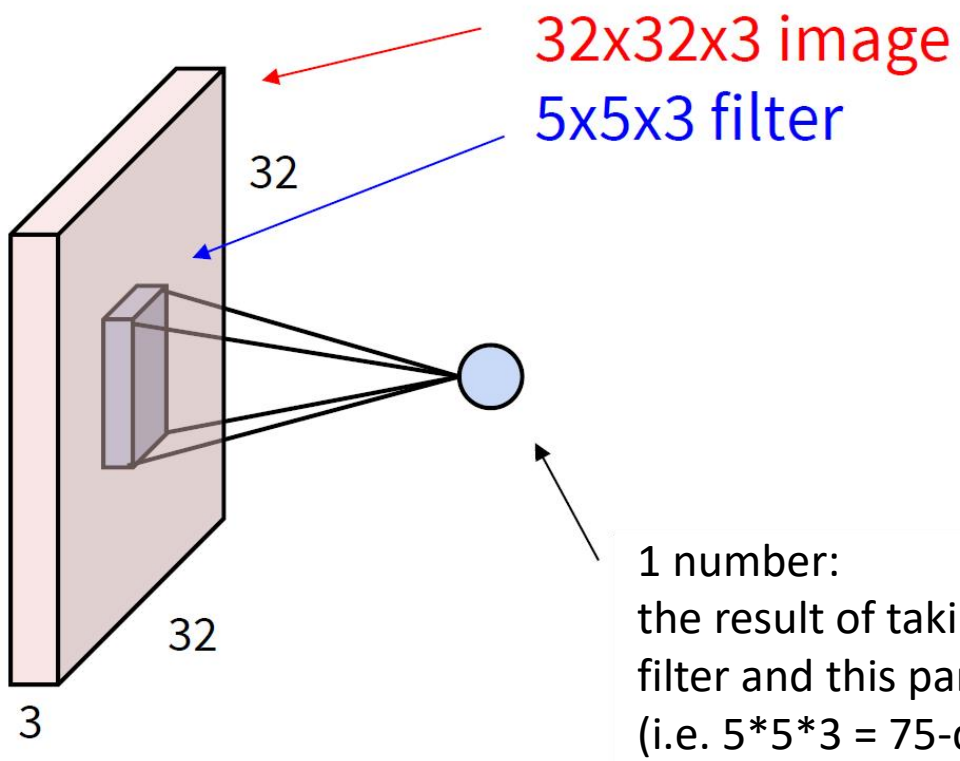
(btw, 1x1 convolution layers make perfect sense)

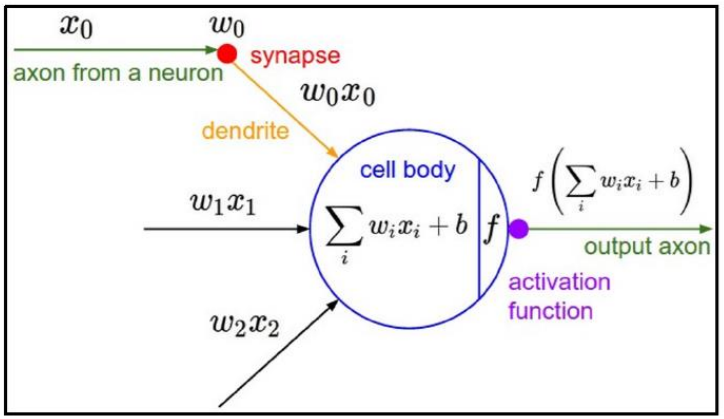


Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

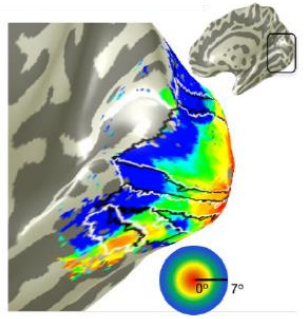
The brain/neuron view of CONV Layer





It's just a neuron with local connectivity...

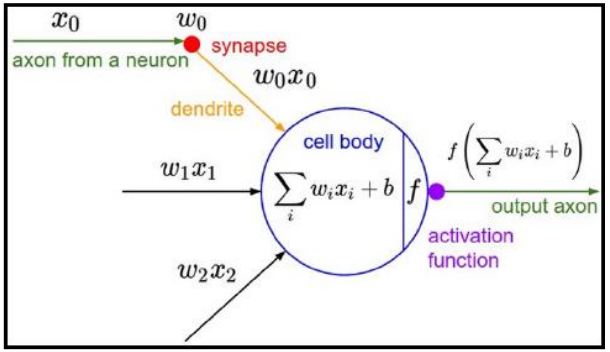
1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

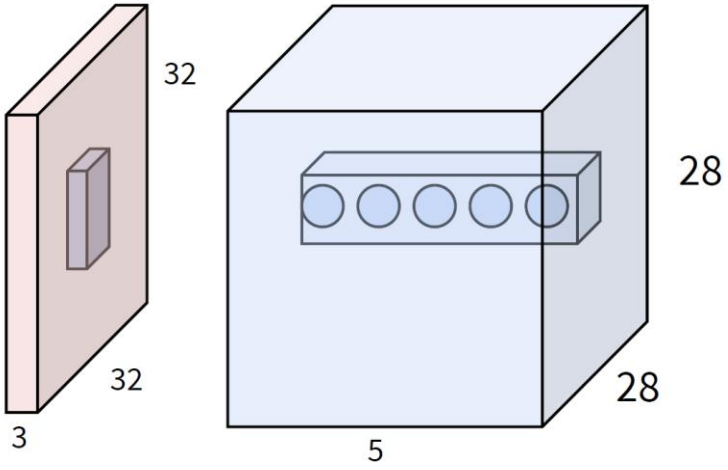


Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

The brain/neuron view of CONV Layer





E.g. with 5 filters, CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

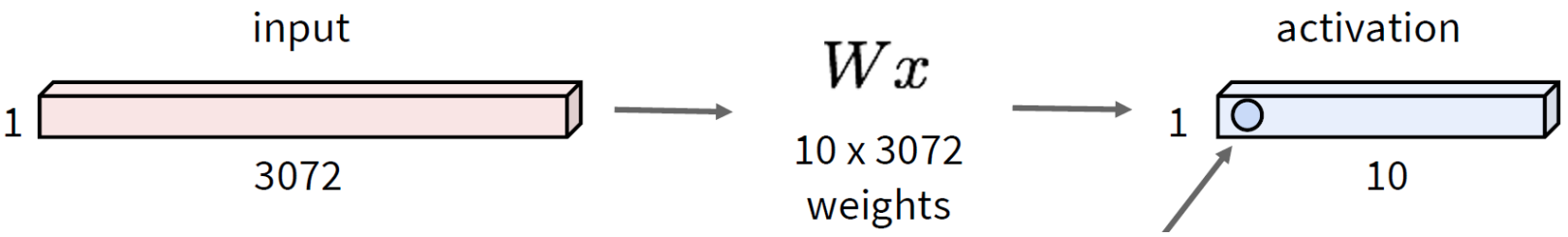
Slide inspiration: Fei-Fei Li, Ehsan Adeli

CONVOLUTION LAYER

Reminder: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

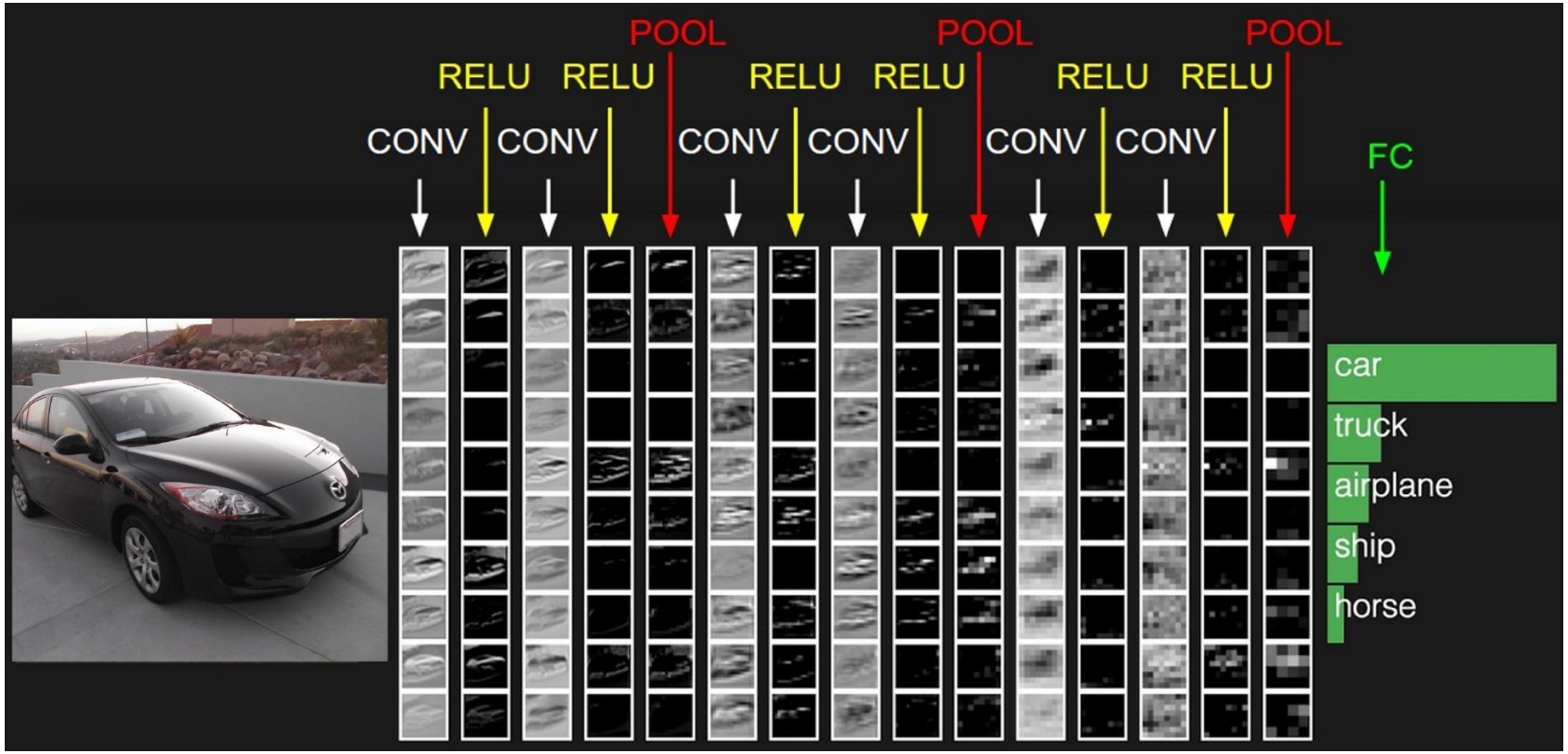
Each neuron looks at the full input volume



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

Slide inspiration: Fei-Fei Li, Ehsan Adeli

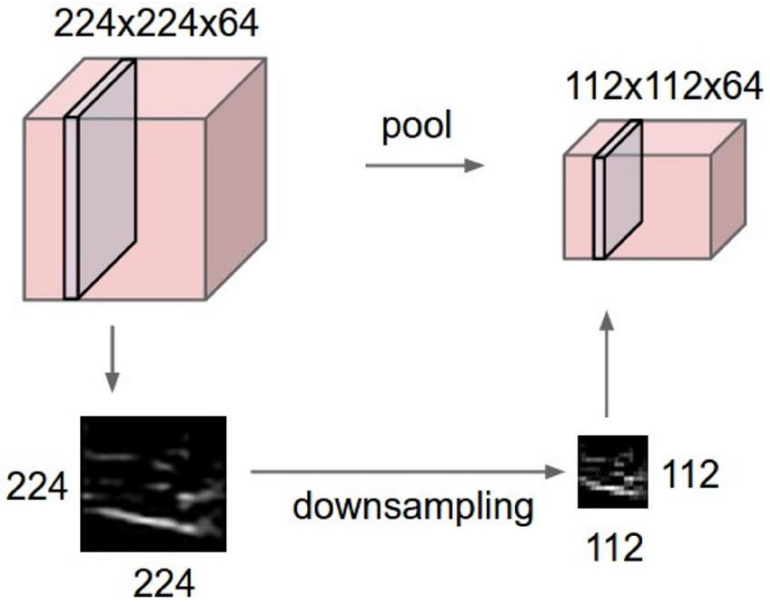
CONVOLUTION LAYER



Slide inspiration: Fei-Fei Li, Ehsan Adeli

POOLING LAYER

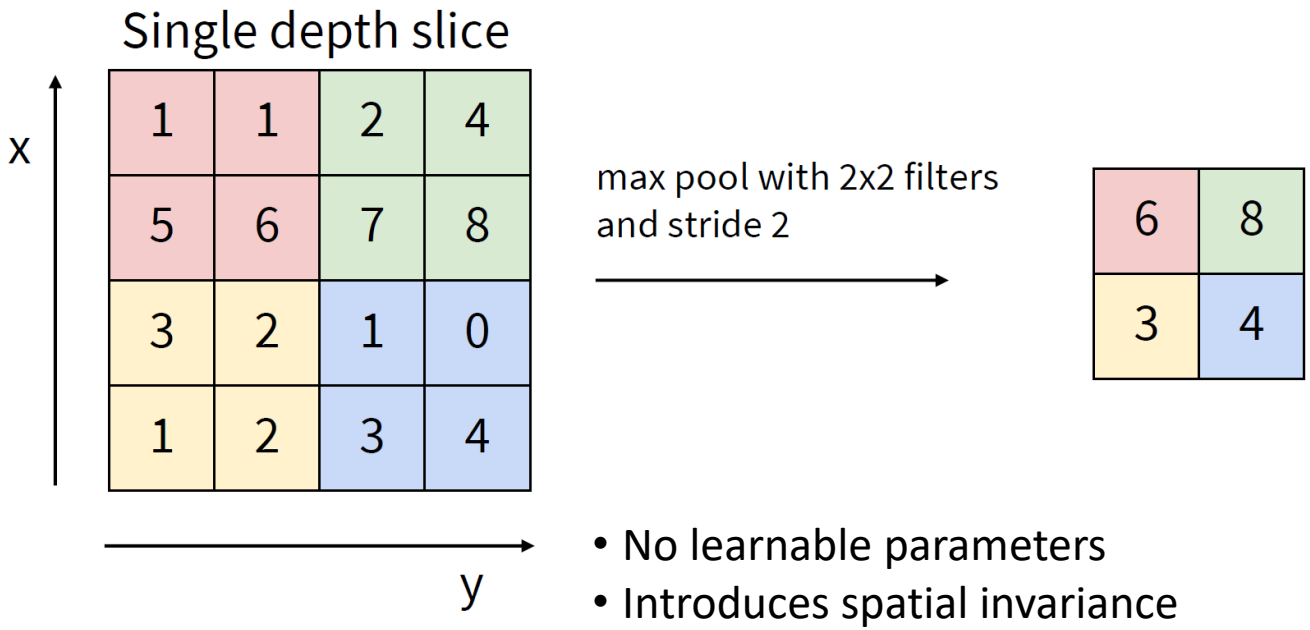
- makes the representations smaller and more manageable
- operates over each activation map independently



Slide inspiration: Fei-Fei Li, Ehsan Adeli

POOLING LAYER

➢ **MAX POOLING**



- No learnable parameters
- Introduces spatial invariance

Slide inspiration: Fei-Fei Li, Ehsan Adeli

POOLING LAYER

➢ **Pooling layer: summary**

Let's assume input is W1 x H1 x C

Conv layer needs 2 hyperparameters:

- The spatial extent F
- The stride S

This will produce an output of W2 x H2 x C where:

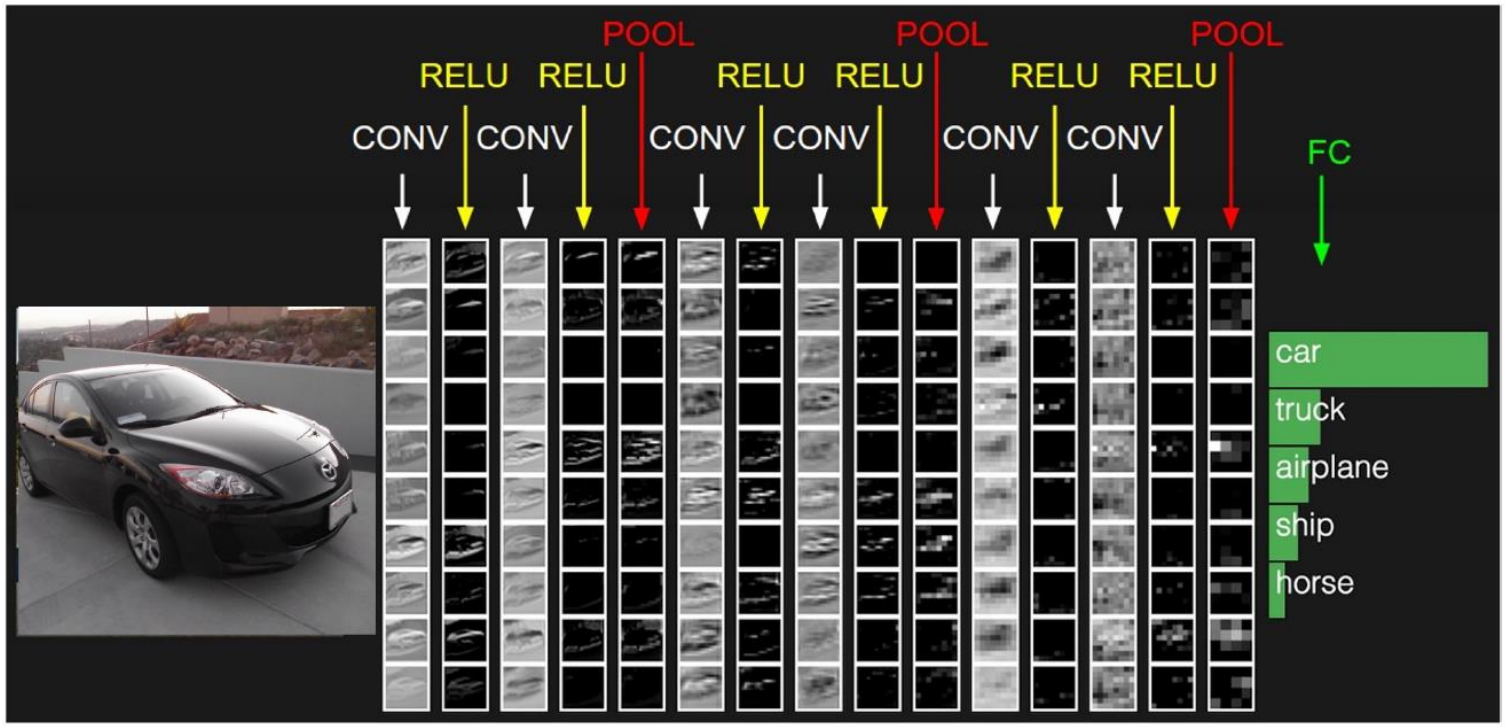
- W2 = (W1 - F)/S + 1
- H2 = (H1 - F)/S + 1

Number of parameters: 0

Slide inspiration: Fei-Fei Li, Ehsan Adeli

FULLY CONNECTED LAYER (FC LAYER)

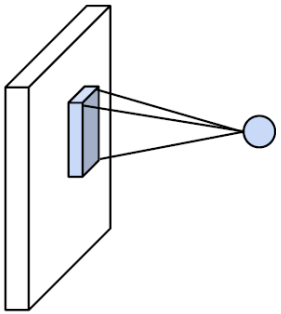
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



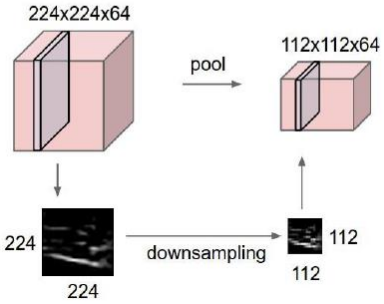
Slide inspiration: Fei-Fei Li, Ehsan Adeli

COMPONENTS OF CNNS

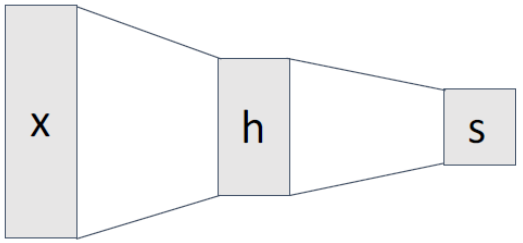
Convolution Layers



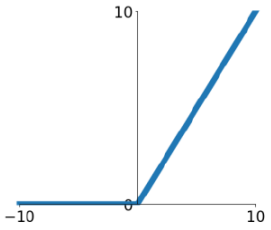
Pooling Layers



Fully-Connected Layers



Activation Function



Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION

- The following could lead to tough optimization:
  - Inputs x are not centered around zero (need large bias)
  - Inputs x have different scaling per-element (entries in W will need to vary a lot)

- **Idea**: force inputs to be "nicely scaled" at each layer!

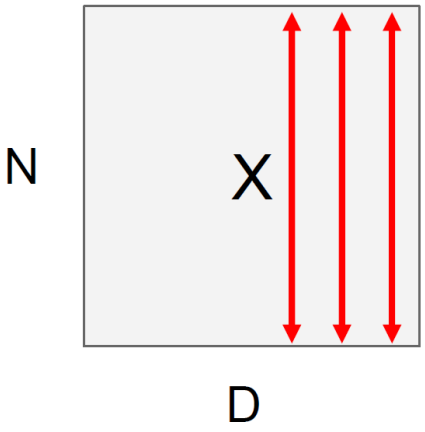
- "you want zero-mean unit-variance activations? just make them so."
  - consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION

Input: $x : N \times D$



$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION

Input: $x : N \times D$

Learnable scale and shift parameters:

$\sigma, \mu: D$

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION: TEST TIME

Estimates depend on minibatch; can't do this at test-time!

Input: $x : N \times D$

Learnable scale and shift parameters:

$\sigma, \mu : D$

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION: TEST TIME

Input: $x : N \times D$

Learnable scale and shift parameters:

$\sigma, \mu : D$

$\mu_j =$ (Running) average of values seen during training

Per-channel mean, shape is D

$\sigma_j^2 =$ (Running) average of values seen during training

Per-channel var, shape is D

During testing batchnorm becomes a linear operator!
Can be fused with the previous fully-connected or conv layer

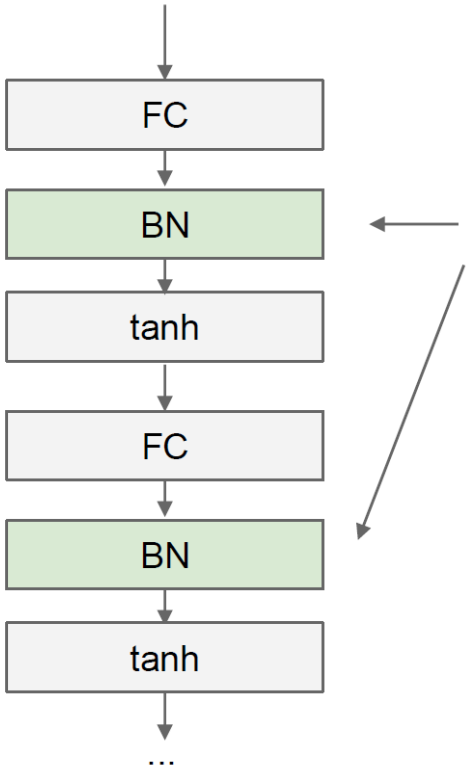
$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION

[Ioffe and Szegedy, 2015]



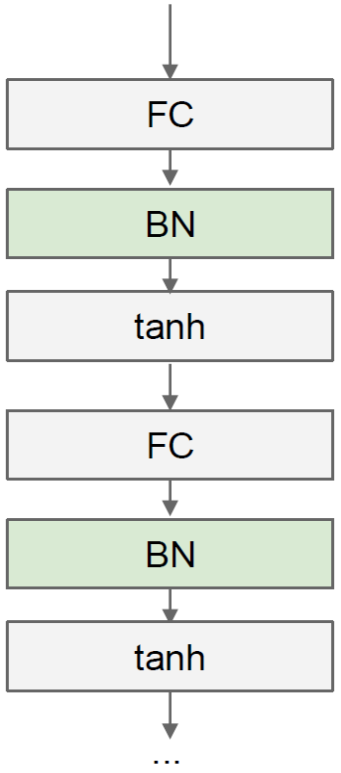
Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION

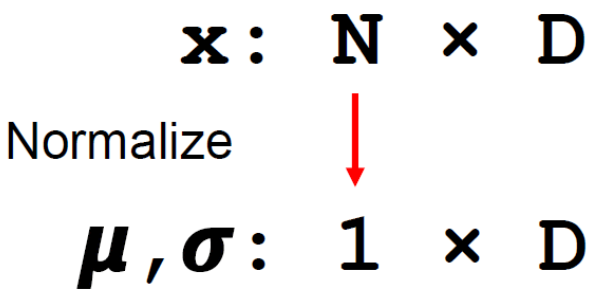
- Makes deep networks **much** easier to train!
- Improves gradient flow
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as a kind of regularization during training



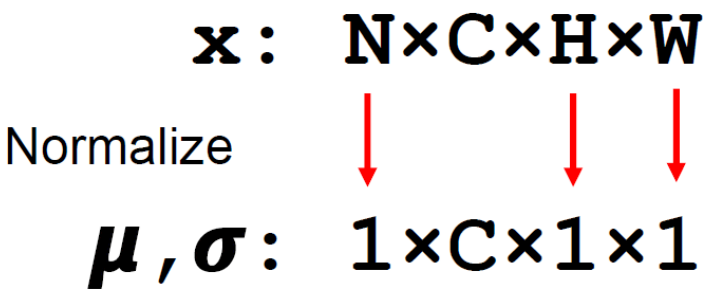
Slide inspiration: Fei-Fei Li, Ehsan Adeli

BATCH NORMALIZATION FOR CONVNETS

Batch Normalization for
fully-connected networks



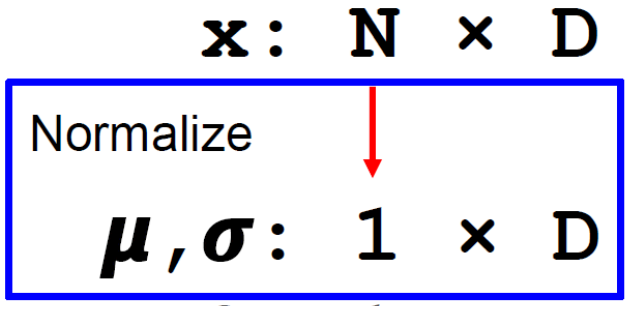
Batch Normalization for
convolutional networks
(Spatial Batchnorm, BatchNorm2D)



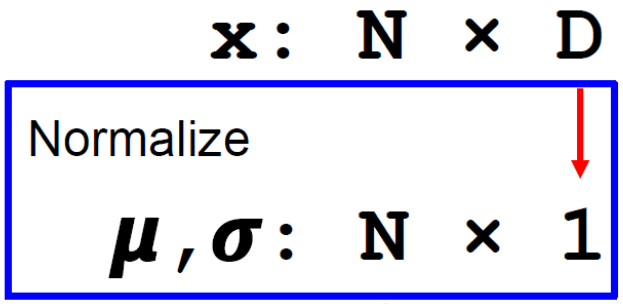
Slide inspiration: Fei-Fei Li, Ehsan Adeli

LAYER NORMALIZATION

Batch Normalization for fully-connected networks



Layer Normalization for fully-connected networks
Same behavior at train and test!

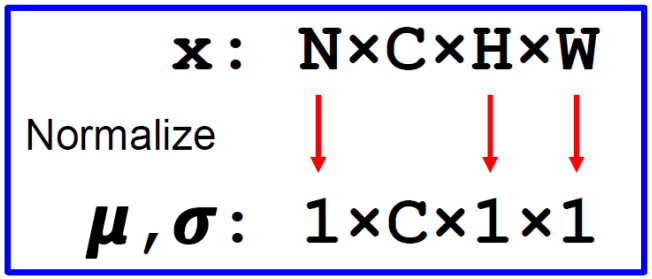


Ba, Kiros, and Hinton, "Layer Normalization", arXiv 2016

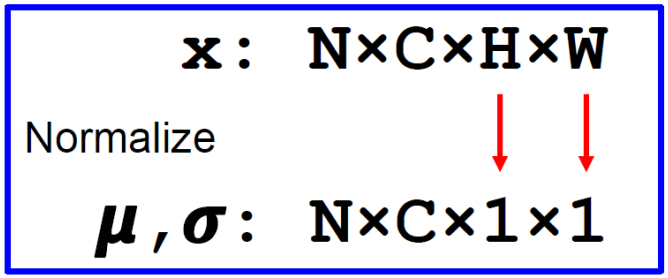
Slide inspiration: Fei-Fei Li, Ehsan Adeli

INSTANCE NORMALIZATION

Batch Normalization for convolutional networks



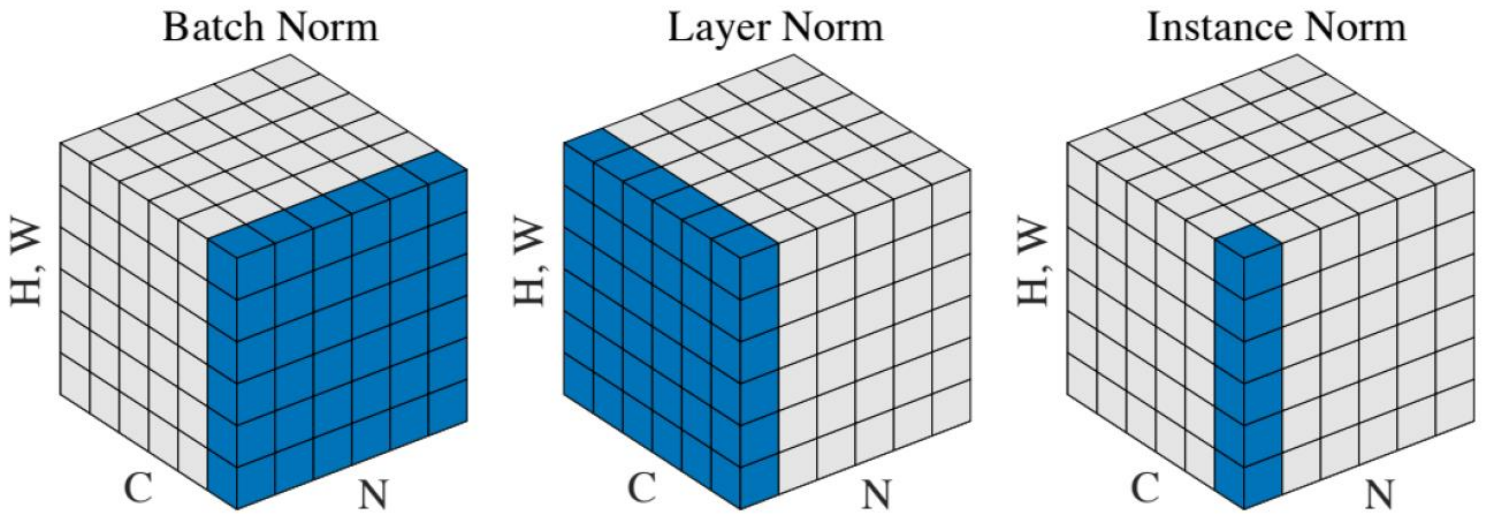
Instance Normalization for convolutional networks
Same behavior at train / test!



Ulyanov et al, Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis, CVPR 2017

Slide inspiration: Fei-Fei Li, Ehsan Adeli

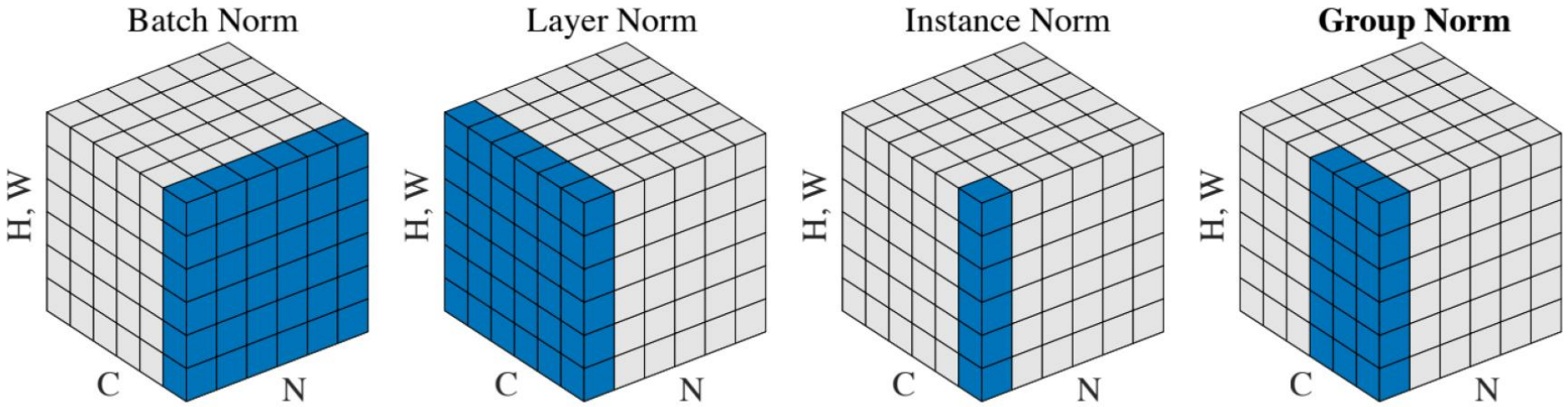
COMPARISON OF NORMALIZATION LAYERS



Wu and He, “Group Normalization”, ECCV 2018

Slide inspiration: Fei-Fei Li, Ehsan Adeli

GROUP NORMALIZATION

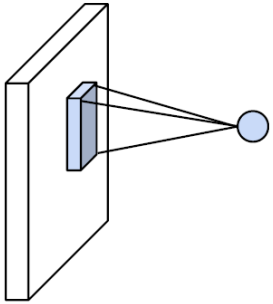


Wu and He, “Group Normalization”, ECCV 2018

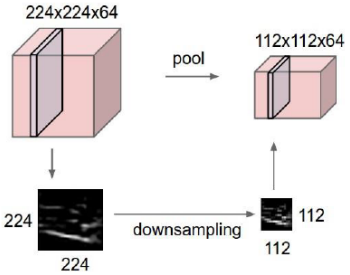
Slide inspiration: Fei-Fei Li, Ehsan Adeli

COMPONENTS OF CNNS

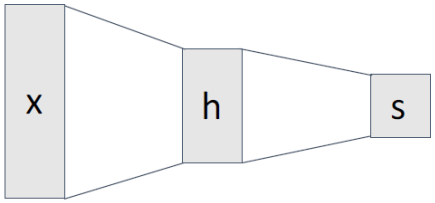
Convolution Layers



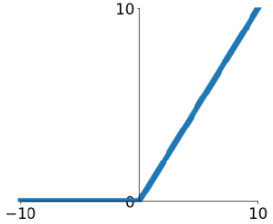
Pooling Layers



Fully-Connected Layers



Activation Function



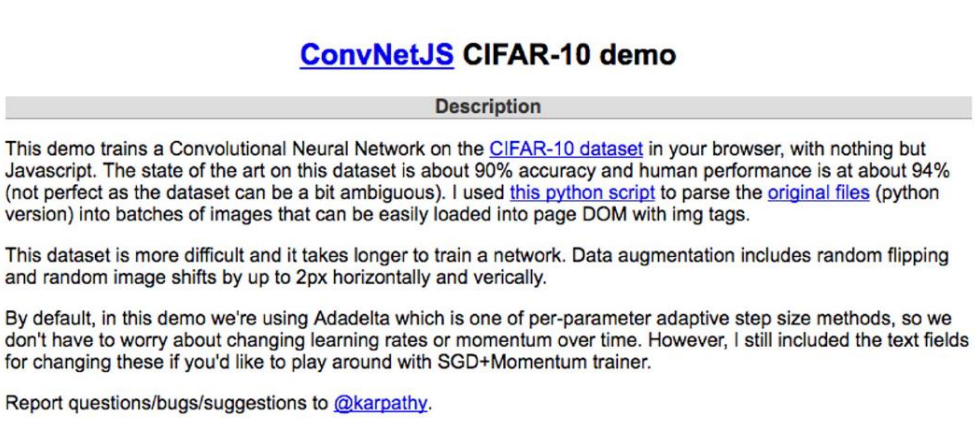
Normalization

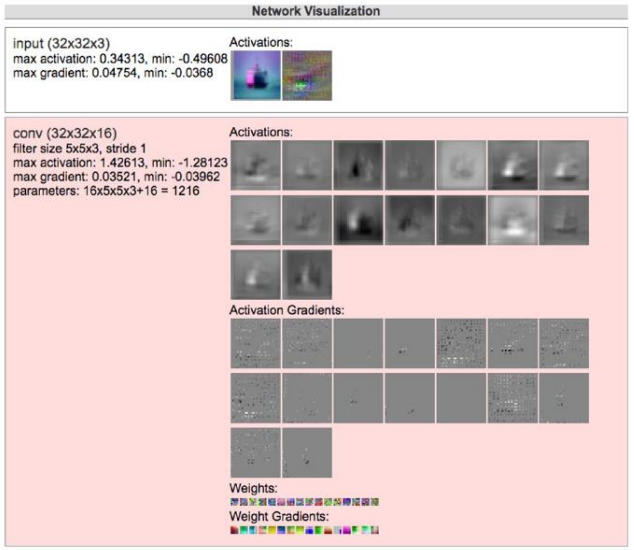
$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Question: How should we put them together?

Slide inspiration: Fei-Fei Li, Ehsan Adeli

[CONVNETJS DEMO: TRAINING ON CIFAR-10]





http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Slide inspiration: Fei-Fei Li, Ehsan Adeli