

# Lecture 5: Image Classification with CNNs

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 1

April 16, 2024

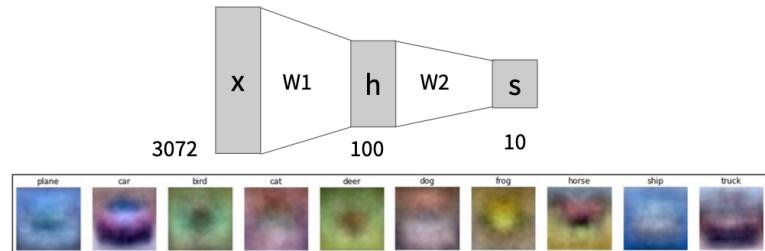
## Last time: Neural Networks

Linear score function:

$$f = Wx$$

2-layer Neural Network

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



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 9

April 16, 2024

## Image Classification: A core task in Computer Vision



This image by [Nikita](#) is  
licensed under [CC BY 2.0](#).

(assume given a set of labels)  
{dog, cat, truck, plane, ...}



cat  
dog  
bird  
deer  
truck

## Pixel space



$$f(x) = Wx$$

Class  
scores



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 15

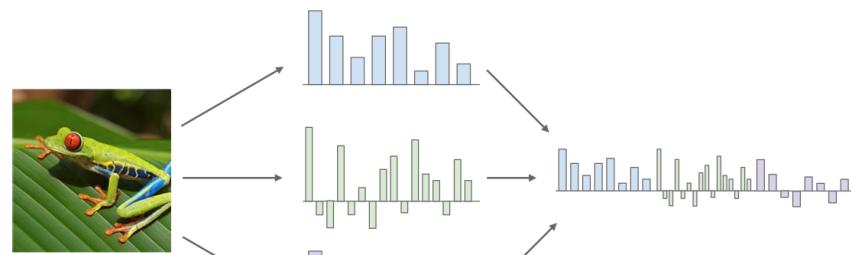
April 16, 2024

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 16

April 16, 2024

## Image Features

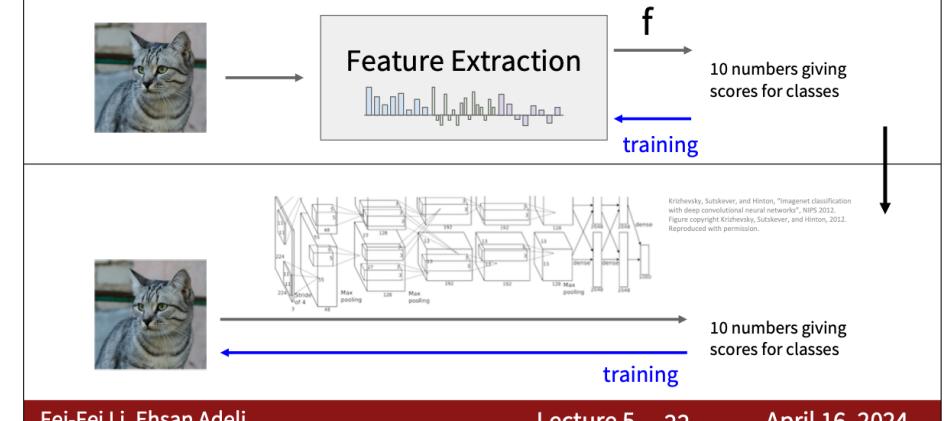


Fei-Fei Li, Ehsan Adeli

Lecture 5 - 21

April 16, 2024

## Image features vs. ConvNets



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 22

April 16, 2024

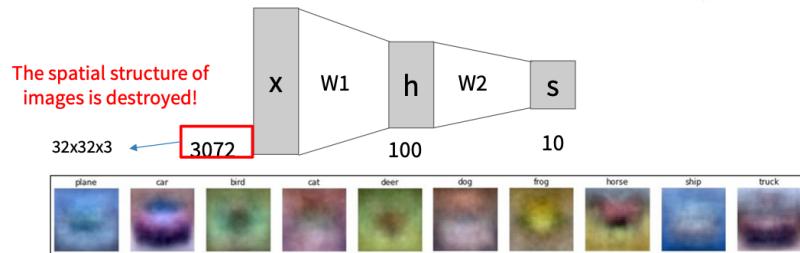
## Last Time: Neural Networks

Linear score function:

$$f = Wx$$

2-layer Neural Network

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



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 23

April 16, 2024

## Convolutional Neural Networks

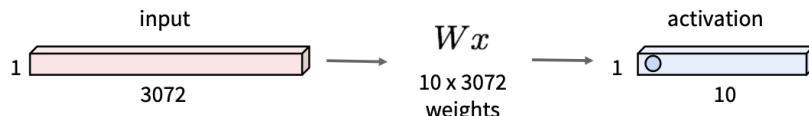
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 45

April 16, 2024

## Recap: Fully Connected Layer

32x32x3 image  $\rightarrow$  stretch to 3072 x 1

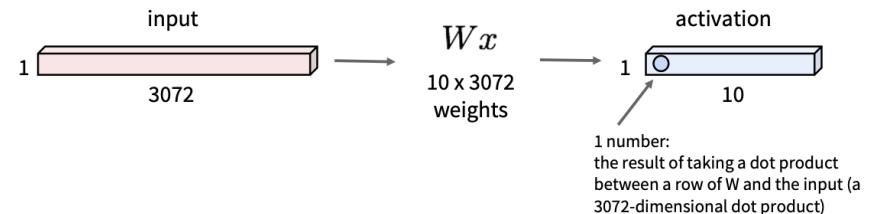


Fei-Fei Li, Ehsan Adeli

Lecture 5 - 46 April 16, 2024

## Fully Connected Layer

32x32x3 image  $\rightarrow$  stretch to 3072 x 1

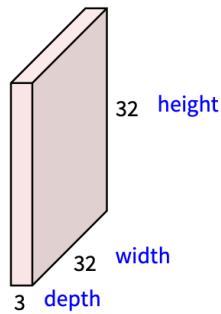


Fei-Fei Li, Ehsan Adeli

Lecture 5 - 47 April 16, 2024

## Convolution Layer

32x32x3 image  $\rightarrow$  preserve spatial structure

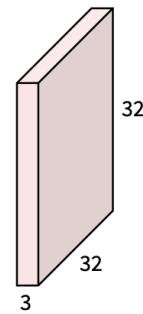


Fei-Fei Li, Ehsan Adeli

Lecture 5 - 48 April 16, 2024

## Convolution Layer

32x32x3 image



5x5x3 filter



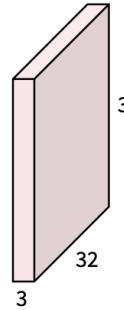
Convolve the filter with the image  
i.e. "slide over the image spatially,  
computing dot products"

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 49 April 16, 2024

## Convolution Layer

32x32x3 image



Filters always extend the full depth of the input volume

5x5x3 filter



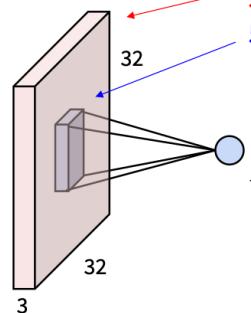
Convolve the filter with the image  
i.e. “slide over the image spatially,  
computing dot products”

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 50 April 16, 2024

## Convolution Layer

32x32x3 image  
5x5x3 filter  $w$



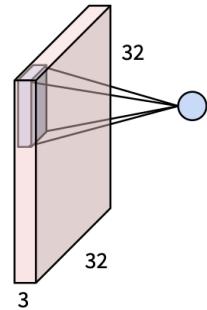
1 number:  
the result of taking a dot product between the  
filter and a small 5x5x3 chunk of the image  
(i.e.  $5 \times 5 \times 3 = 75$ -dimensional dot product + bias)

$$w^T x + b$$

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 51 April 16, 2024

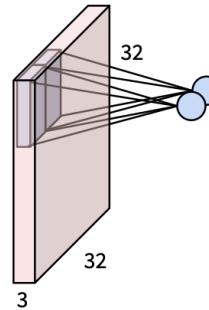
## Convolution Layer



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 52 April 16, 2024

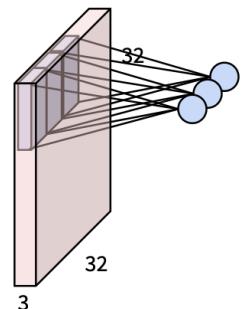
## Convolution Layer



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 53 April 16, 2024

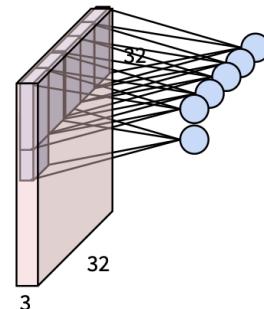
## Convolution Layer



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 54 April 16, 2024

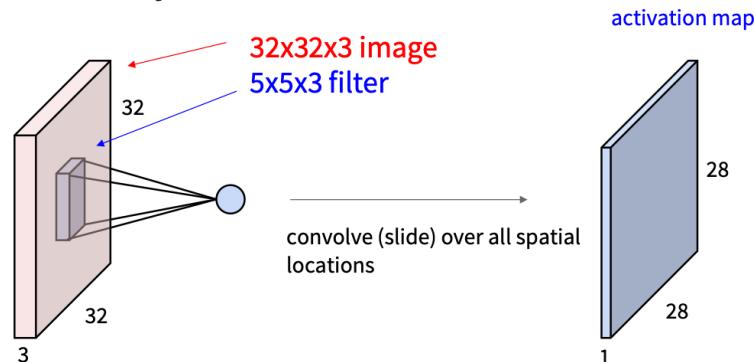
## Convolution Layer



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 55 April 16, 2024

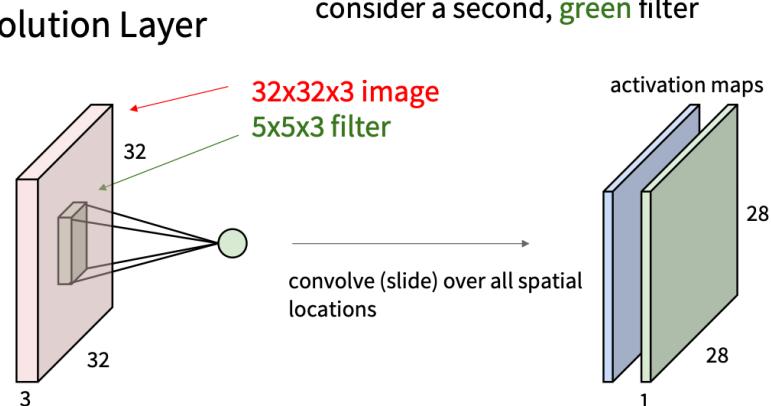
## Convolution Layer



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 56 April 16, 2024

## Convolution Layer

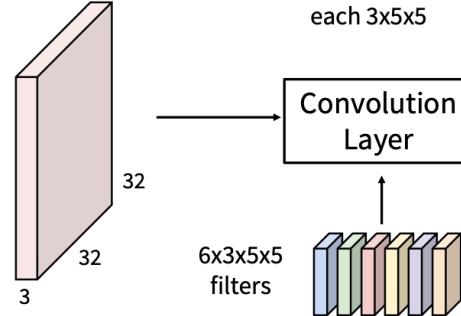


Fei-Fei Li, Ehsan Adeli

Lecture 5 - 57 April 16, 2024

## Convolution Layer

3x32x32 image



Consider 6 filters,  
each 3x5x5

6 activation maps,  
each 1x28x28

Stack activations to get a  
6x28x28 output image!

Slide inspiration: Justin Johnson

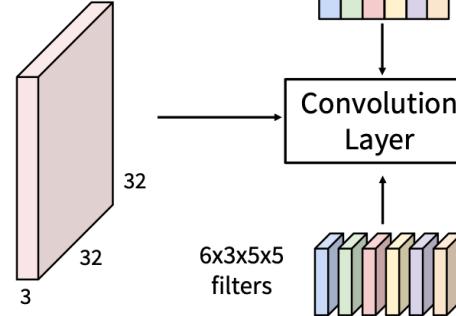
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 58

April 16, 2024

## Convolution Layer

3x32x32 image



Also 6-dim bias vector:

6 activation maps,  
each 1x28x28

Stack activations to get a  
6x28x28 output image!

Slide inspiration: Justin Johnson

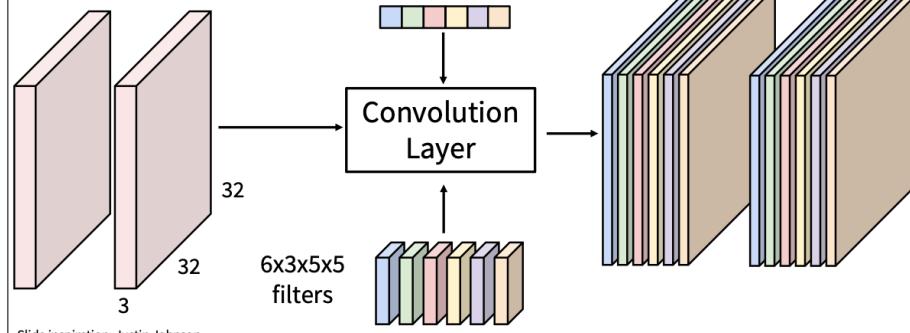
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 59

April 16, 2024

## Convolution Layer

2x3x32x32  
Batch of images



Also 6-dim bias vector:

2x6x28x28  
Batch of outputs

Slide inspiration: Justin Johnson

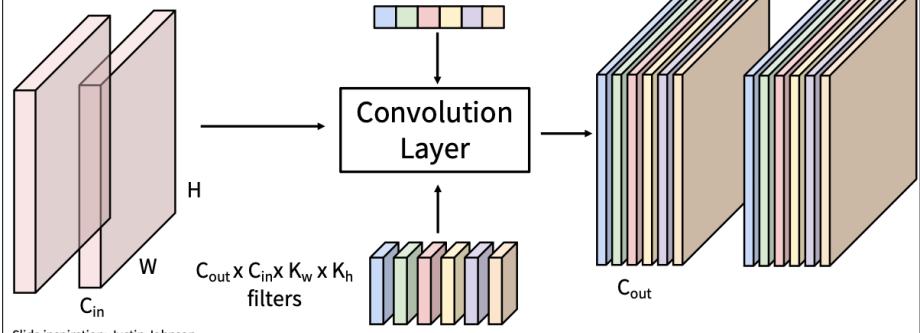
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 61

April 16, 2024

## Convolution Layer

N x C<sub>in</sub> x H x W  
Batch of images



Also C<sub>out</sub>-dim bias vector:

N x C<sub>out</sub> x H' x W'  
Batch of outputs

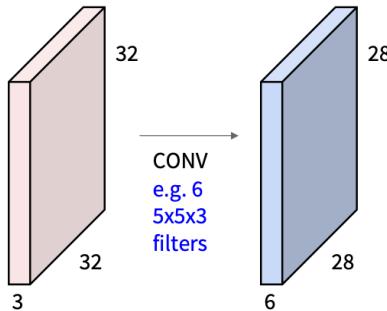
Slide inspiration: Justin Johnson

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 62

April 16, 2024

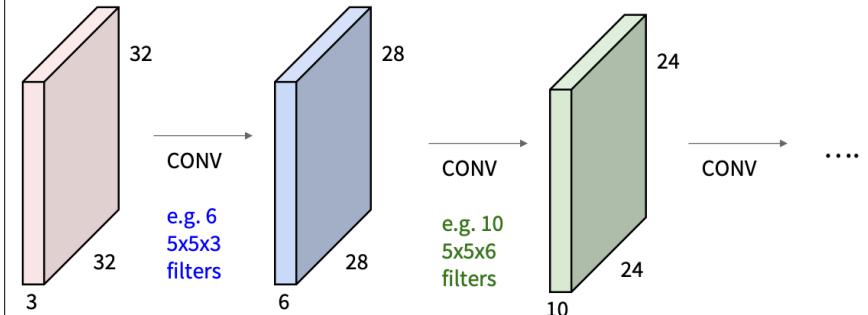
Preview: ConvNet is a sequence of Convolution Layers



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 63 April 16, 2024

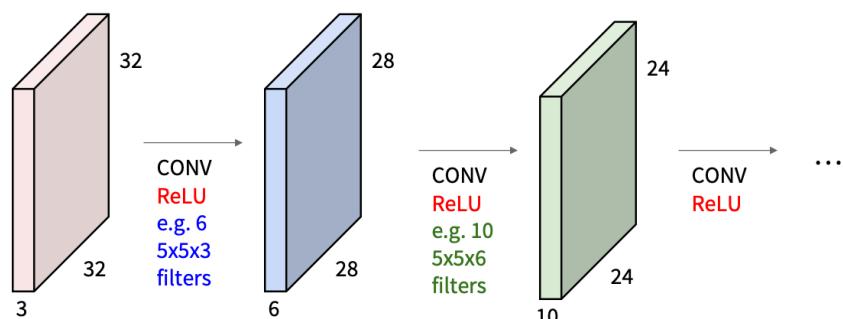
Preview: ConvNet is a sequence of Convolution Layers



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 64 April 16, 2024

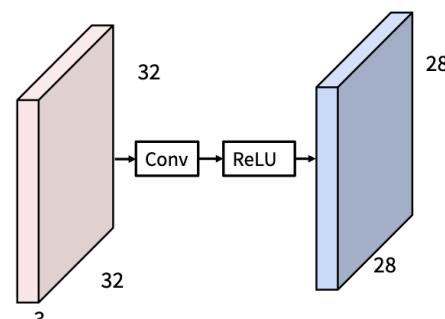
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



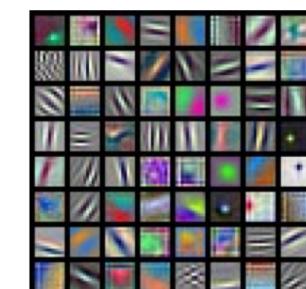
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 65 April 16, 2024

Preview: What do convolutional filters learn?



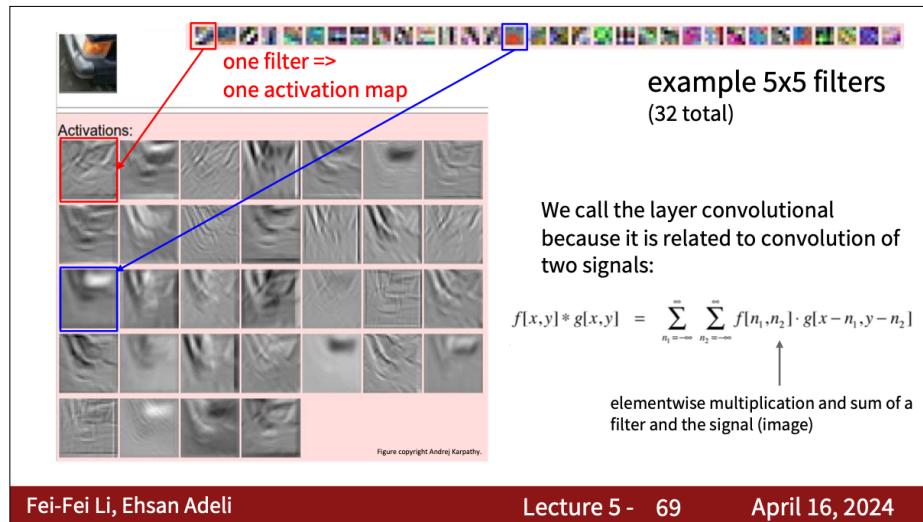
First-layer conv filters: local image templates  
(Often learns oriented edges, opposing colors)



AlexNet: 64 filters, each 3x11x11

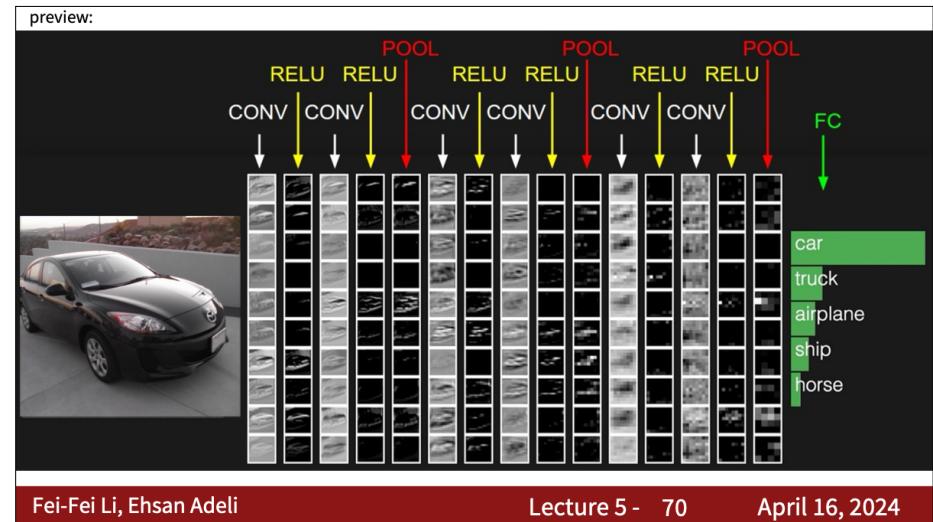
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 68 April 16, 2024



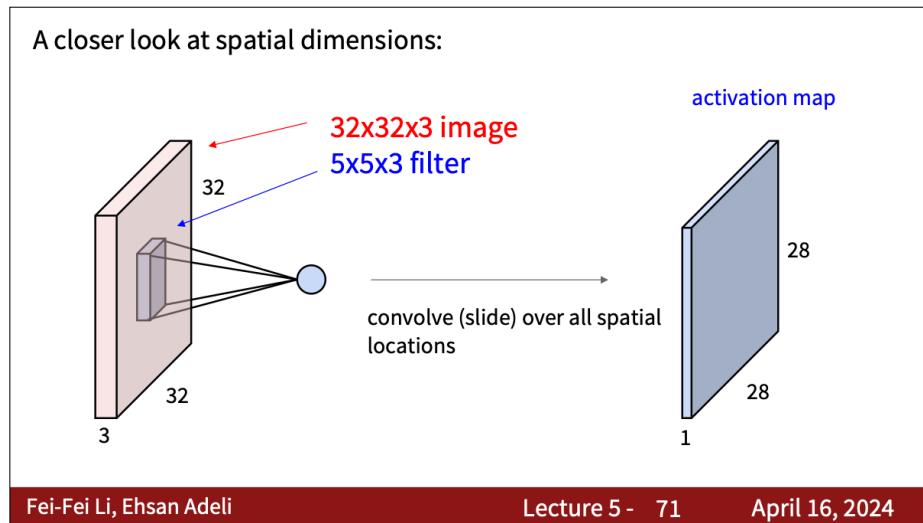
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 69 April 16, 2024



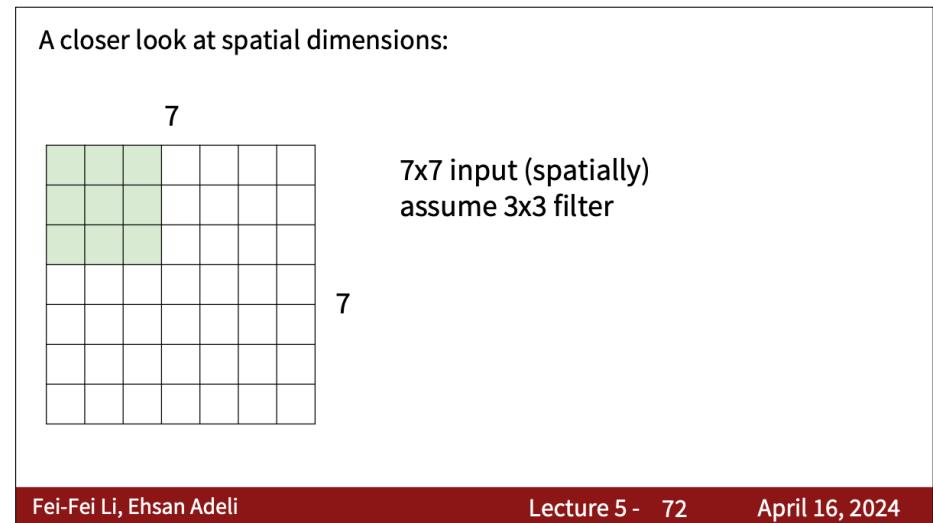
Fei-Fei Li, Ehsan Adeli

Lecture 5 - 70 April 16, 2024



Fei-Fei Li, Ehsan Adeli

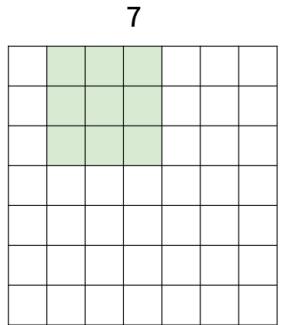
Lecture 5 - 71 April 16, 2024



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 72 April 16, 2024

A closer look at spatial dimensions:

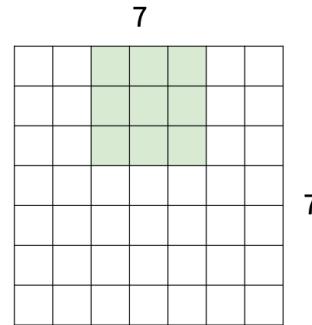


7x7 input (spatially)  
assume 3x3 filter

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 73 April 16, 2024

A closer look at spatial dimensions:



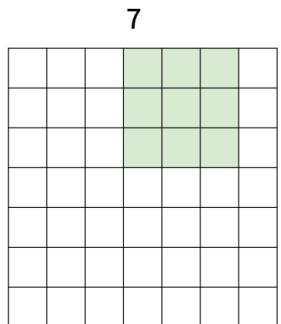
7x7 input (spatially)  
assume 3x3 filter

7

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 74 April 16, 2024

A closer look at spatial dimensions:

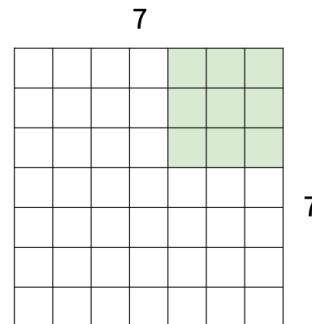


7x7 input (spatially)  
assume 3x3 filter

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 75 April 16, 2024

A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter

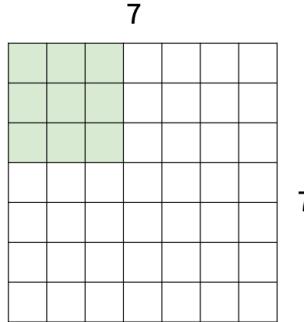
=> 5x5 output

7

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 76 April 16, 2024

A closer look at spatial dimensions:

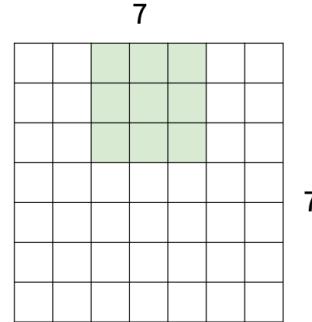


7x7 input (spatially)  
assume 3x3 filter  
applied with stride 2

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 77 April 16, 2024

A closer look at spatial dimensions:

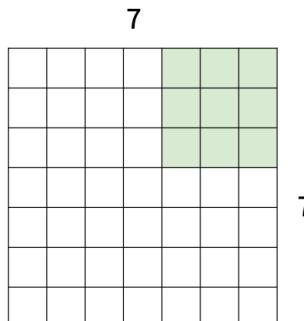


7x7 input (spatially)  
assume 3x3 filter  
applied with stride 2

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 78 April 16, 2024

A closer look at spatial dimensions:

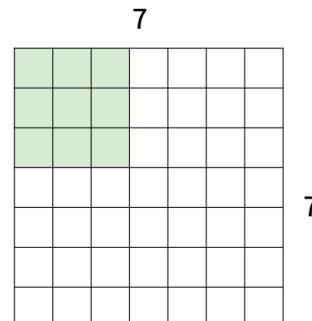


7x7 input (spatially)  
assume 3x3 filter  
applied with stride 2  
=> 3x3 output!

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 79 April 16, 2024

A closer look at spatial dimensions:

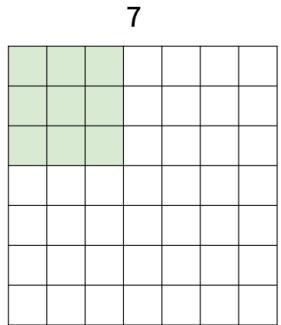


7x7 input (spatially)  
assume 3x3 filter  
applied with stride 3?

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 80 April 16, 2024

A closer look at spatial dimensions:



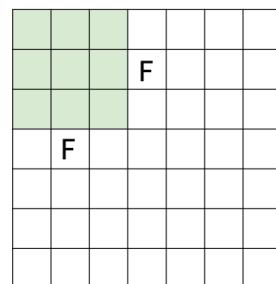
7x7 input (spatially)  
assume 3x3 filter  
applied with stride 3?

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

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 81 April 16, 2024

N



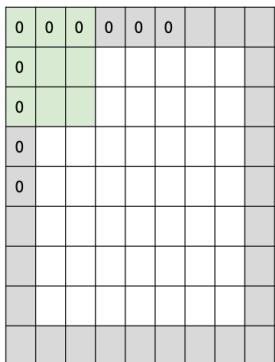
Output size:  
 $(N - F) / \text{stride} + 1$

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

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 82 April 16, 2024

In practice: Common to zero pad the border



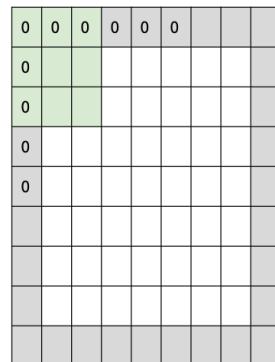
e.g. input 7x7  
3x3 filter, applied with stride 1  
pad with 1 pixel border => what is the output?

(recall):  
 $(N - F) / \text{stride} + 1$

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 83 April 16, 2024

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?

7x7 output!

(recall):  
 $(N + 2P - F) / \text{stride} + 1$

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 84 April 16, 2024

## In practice: Common to zero pad the border

0	0	0	0	0	0	0
0						
0						
0						
0						

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)

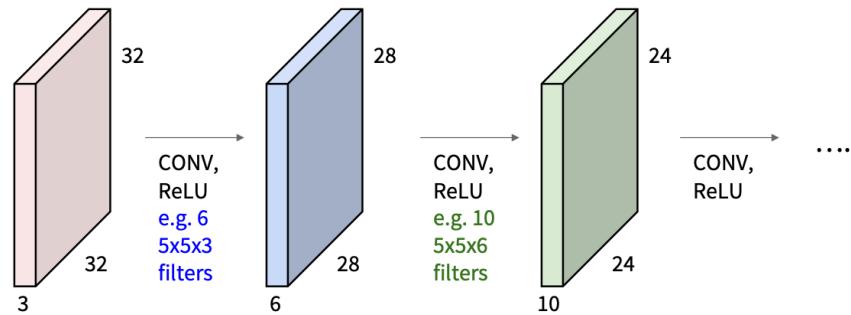
e.g.  $F=3 \Rightarrow$  zero pad with 1

$F=5 \Rightarrow$  zero pad with 2

$F=7 \Rightarrow$  zero pad with 3

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.



## Convolution layer: summary

Let's assume input is  $W_1 \times H_1 \times C$

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  $W_2 \times H_2 \times K$

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

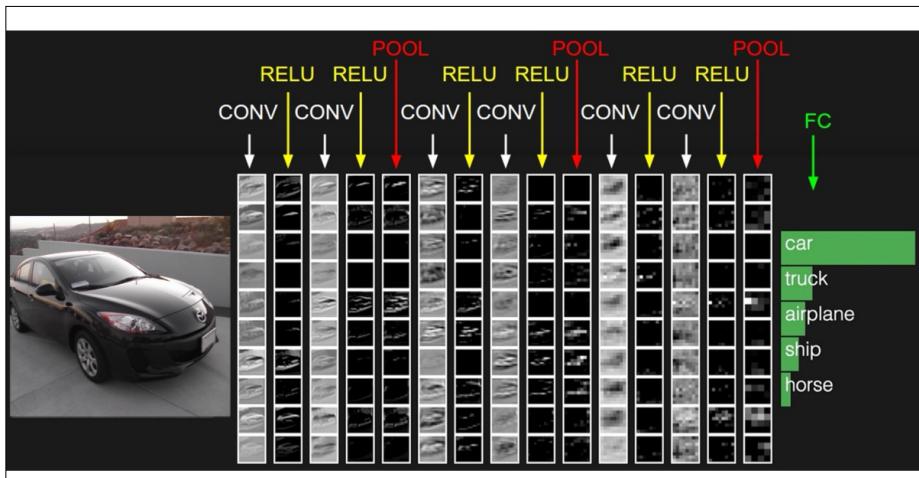
## Example: CONV layer in PyTorch

```
Conv2d
CLASS torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)
Applies a 2D convolution over an input signal composed of several input planes.
In the simplest case, the output value of the layer with input size  $(N, C_{in}, H, W)$  and output  $(N, C_{out}, H_{out}, W_{out})$  can be precisely described as

$$out(N_i, C_{out,j}) = bias(C_{out,j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out,j}, k) * input(N_i, k)$$

where  $*$  is the valid 2D cross-correlation operator,  $N$  is a batch size,  $C$  denotes a number of channels,  $H$  is a height of input planes in pixels, and  $W$  is width in pixels.
• stride controls the stride for the spatial cross-correlation, a single number or a tuple.
• padding controls the amount of implicit zero-paddings on both sides for padding: number of points for each side.
• dilation controls the spacing between the kernel points, also known as the a trous algorithm. It is harder to describe, but this link has a nice visualization of what dilation does.
• groups controls the connections between inputs and outputs. in_channels and out_channels must both be divisible by groups. For example:
    • At groups=1, all inputs are convolved to all outputs.
    • At groups=2, the operation becomes equivalent to having two conv layers side by side, each seeing half the input channels, and producing half the output channels, and both subsequently concatenated.
    • At groups = in_channels, each input channel is convolved with its own set of filters, of size  $\left\lceil \frac{C_{out}}{C_{in}} \right\rceil$ .
The parameters kernel_size, stride, padding, dilation can either be:
• a single int - in which case the same value is used for the height and width dimensions;
• a tuple of two ints - in which case, the first int is used for the height dimension, and the second for the width dimension.
```

[PyTorch](#) is licensed under [BSD 3-clause](#).

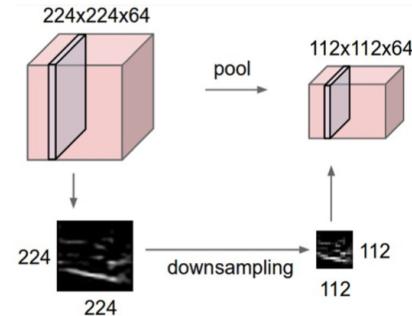


Fei-Fei Li, Ehsan Adeli

Lecture 5 - 106 April 16, 2024

## Pooling layer

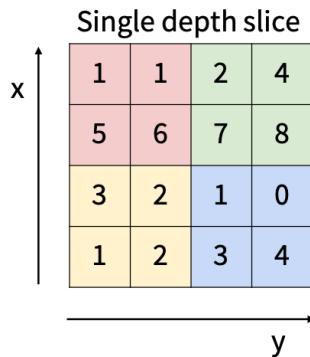
- makes the representations smaller and more manageable
- operates over each activation map independently



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 107 April 16, 2024

## MAX POOLING

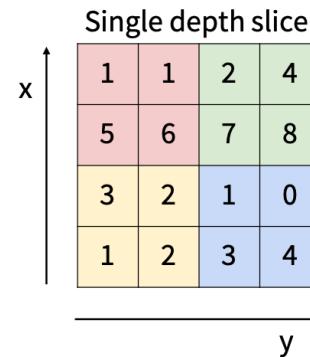


6	8
3	4

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 108 April 16, 2024

## MAX POOLING



6	8
3	4

- No learnable parameters
- Introduces spatial invariance

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 109 April 16, 2024

## Pooling layer: summary

Let's assume input is  $W_1 \times H_1 \times C$

Conv layer needs 2 hyperparameters:

- The spatial extent F
- The stride S

This will produce an output of  $W_2 \times H_2 \times C$  where:

- $W_2 = (W_1 - F)/S + 1$
- $H_2 = (H_1 - F)/S + 1$

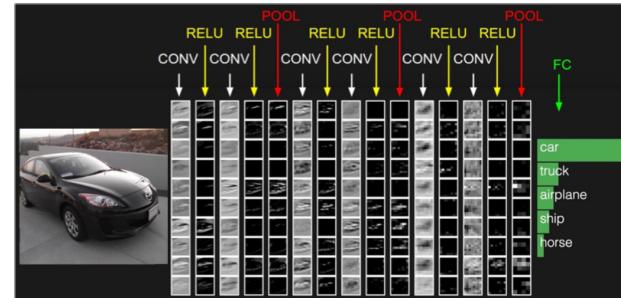
Number of parameters: 0

Fei-Fei Li, Ehsan Adeli

Lecture 5 - 110 April 16, 2024

## Fully Connected Layer (FC layer)

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



Fei-Fei Li, Ehsan Adeli

Lecture 5 - 111 April 16, 2024

## Summary

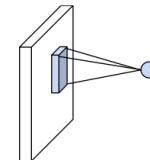
- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Historically architectures looked like  
[(CONV-RELU)\*N-POOL?]\*M-(FC-RELU)\*K,SOFTMAX  
where N is usually up to ~5, M is large, 0 <= K <= 2.
- But recent advances such as ResNet/GoogLeNet have challenged this paradigm

Fei-Fei Li, Ehsan Adeli

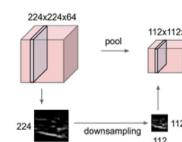
Lecture 5 - 113 April 16, 2024

## Components of CNNs

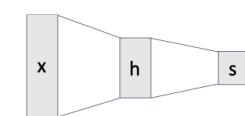
### Convolution Layers



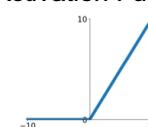
### Pooling Layers



### Fully-Connected Layers



### Activation Function



### Normalization

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

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 10 April 17, 2024

## Batch Normalization

Consider a single layer  $y = Wx$

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!

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 11

April 17, 2024

[Ioffe and Szegedy, 2015]

## Batch Normalization

“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:

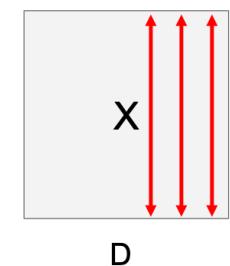
$$\hat{x}^{(k)} = \frac{x^{(k)} - \text{E}[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}$$

this is a vanilla  
differentiable function...

## Batch Normalization

[Ioffe and Szegedy, 2015]

**Input:**  $x : N \times D$



$$\mu_j = \frac{1}{N} \sum_{i=1}^N x_{i,j} \quad \text{Per-channel mean, shape is } D$$

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^N (x_{i,j} - \mu_j)^2 \quad \text{Per-channel var, shape is } D$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}} \quad \text{Normalized } x, \text{ Shape is } N \times D$$

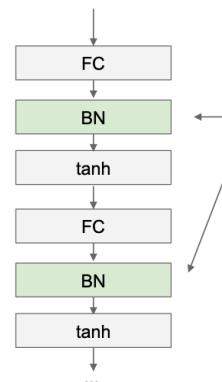
Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 13

April 17, 2024

## Batch Normalization

[Ioffe and Szegedy, 2015]



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

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

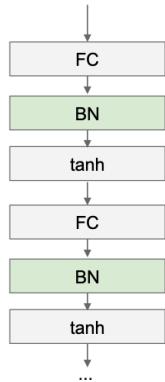
Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 18

April 17, 2024

## Batch Normalization

[Ioffe and Szegedy, 2015]



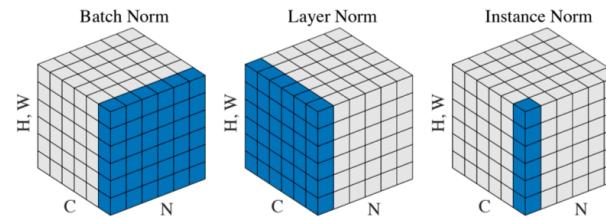
- Makes deep networks **much** easier to train!
- Improves gradient flow
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as a kind of regularization during training
- Behaves differently during training and testing: this is a very common source of bugs!

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 19

April 17, 2024

## Comparison of Normalization Layers



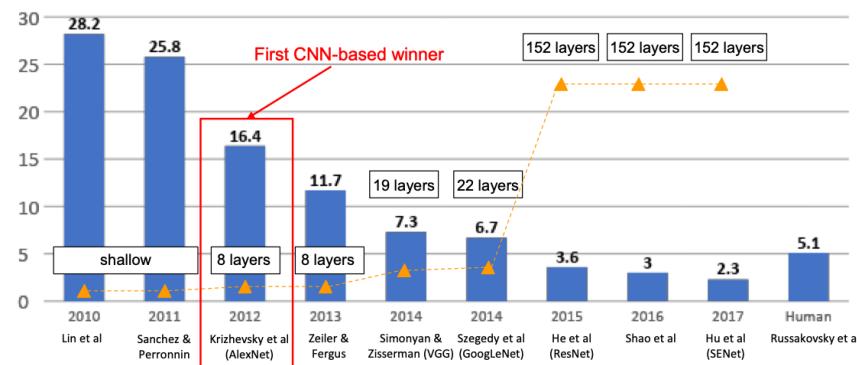
Wu and He, "Group Normalization", ECCV 2018

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 23

April 17, 2024

## ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 28

April 17, 2024

## Case Study: AlexNet

[Krizhevsky et al. 2012]

### Architecture:

CONV1  
MAX POOL1  
NORM1  
CONV2  
MAX POOL2  
NORM2  
CONV3  
CONV4  
CONV5  
Max POOL3  
FC6  
FC7  
FC8

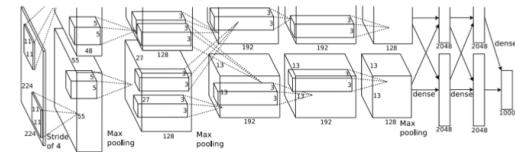


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 29

April 17, 2024

## Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT  
 [55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0  
 [27x27x96] MAX POOL1: 3x3 filters at stride 2  
 [27x27x96] NORM1: Normalization layer  
 [27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2  
 [13x13x256] MAX POOL2: 3x3 filters at stride 2  
 [13x13x256] NORM2: Normalization layer  
 [13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1  
 [13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1  
 [13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1  
 [6x6x256] MAX POOL3: 3x3 filters at stride 2  
 [4096] FC6: 4096 neurons  
 [4096] FC7: 4096 neurons  
 [1000] FC8: 1000 neurons (class scores)

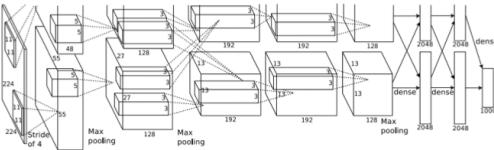


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

## Case Study: VGGNet

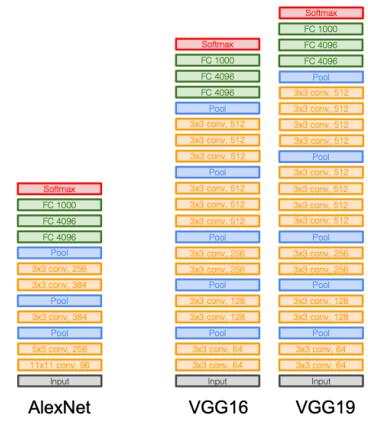
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)  
 -> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1  
 and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13  
 (ZFNet)  
 -> 7.3% top 5 error in ILSVRC'14



Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 39

April 17, 2024

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 42

April 17, 2024

INPUT: [224x224x3] memory: 224\*224\*3=150K params: 0 (not counting biases)  
 CONV3-64: [224x224x64] memory: 224\*224\*64=3.2M params: (3\*3\*3)\*64 = 1,728  
 CONV3-64: [224x224x64] memory: 224\*224\*64=3.2M params: (3\*3\*64)\*64 = 36,864  
 POOL2: [112x112x64] memory: 112\*112\*64=800K params: 0  
 CONV3-128: [112x112x128] memory: 112\*112\*128=1.6M params: (3\*3\*64)\*128 = 73,728  
 CONV3-128: [112x112x128] memory: 112\*112\*128=1.6M params: (3\*3\*128)\*128 = 147,456  
 POOL2: [56x56x128] memory: 56\*56\*128=400K params: 0  
 CONV3-256: [56x56x256] memory: 56\*56\*256=800K params: (3\*3\*128)\*256 = 294,912  
 CONV3-256: [56x56x256] memory: 56\*56\*256=800K params: (3\*3\*256)\*256 = 589,824  
 CONV3-256: [56x56x256] memory: 56\*56\*256=800K params: (3\*3\*256)\*256 = 589,824  
 POOL2: [28x28x256] memory: 28\*28\*256=200K params: 0  
 CONV3-512: [28x28x512] memory: 28\*28\*512=400K params: (3\*3\*256)\*512 = 1,179,648  
 CONV3-512: [28x28x512] memory: 28\*28\*512=400K params: (3\*3\*512)\*512 = 2,359,296  
 CONV3-512: [28x28x512] memory: 28\*28\*512=400K params: (3\*3\*512)\*512 = 2,359,296  
 POOL2: [14x14x512] memory: 14\*14\*512=100K params: 0  
 CONV3-512: [14x14x512] memory: 14\*14\*512=100K params: (3\*3\*512)\*512 = 2,359,296  
 CONV3-512: [14x14x512] memory: 14\*14\*512=100K params: (3\*3\*512)\*512 = 2,359,296  
 CONV3-512: [14x14x512] memory: 14\*14\*512=100K params: (3\*3\*512)\*512 = 2,359,296  
 POOL2: [7x7x512] memory: 7\*7\*512=25K params: 0  
 FC: [1x1x4096] memory: 4096 params: 7\*7\*512\*4096 = 102,760,448  
 FC: [1x1x4096] memory: 4096 params: 4096\*4096 = 16,777,216  
 FC: [1x1x1000] memory: 1000 params: 4096\*1000 = 4,096,000  
 TOTAL memory: 24M \* 4 bytes ~ 96MB / image (for a forward pass)  
 TOTAL params: 138M parameters

INPUT: [224x224x3] memory: 224\*224\*3=150K params: 0 (not counting biases)  
 CONV3-64: [224x224x64] memory: 224\*224\*64=3.2M params: (3\*3\*3)\*64 = 1,728  
 CONV3-64: [224x224x64] memory: 224\*224\*64=3.2M params: (3\*3\*64)\*64 = 36,864  
 POOL2: [112x112x64] memory: 112\*112\*64=800K params: 0  
 CONV3-128: [112x112x128] memory: 112\*112\*128=1.6M params: (3\*3\*64)\*128 = 73,728  
 CONV3-128: [112x112x128] memory: 112\*112\*128=1.6M params: (3\*3\*128)\*128 = 147,456  
 POOL2: [56x56x128] memory: 56\*56\*128=400K params: 0  
 CONV3-256: [56x56x256] memory: 56\*56\*256=800K params: (3\*3\*128)\*256 = 294,912  
 CONV3-256: [56x56x256] memory: 56\*56\*256=800K params: (3\*3\*256)\*256 = 589,824  
 CONV3-256: [56x56x256] memory: 56\*56\*256=800K params: (3\*3\*256)\*256 = 589,824  
 POOL2: [28x28x256] memory: 28\*28\*256=200K params: 0  
 CONV3-512: [28x28x512] memory: 28\*28\*512=400K params: (3\*3\*256)\*512 = 1,179,648  
 CONV3-512: [28x28x512] memory: 28\*28\*512=400K params: (3\*3\*512)\*512 = 2,359,296  
 CONV3-512: [28x28x512] memory: 28\*28\*512=400K params: (3\*3\*512)\*512 = 2,359,296  
 POOL2: [14x14x512] memory: 14\*14\*512=100K params: 0  
 CONV3-512: [14x14x512] memory: 14\*14\*512=100K params: (3\*3\*512)\*512 = 2,359,296  
 CONV3-512: [14x14x512] memory: 14\*14\*512=100K params: (3\*3\*512)\*512 = 2,359,296  
 CONV3-512: [14x14x512] memory: 14\*14\*512=100K params: (3\*3\*512)\*512 = 2,359,296  
 POOL2: [7x7x512] memory: 7\*7\*512=25K params: 0  
 FC: [1x1x4096] memory: 4096 params: 7\*7\*512\*4096 = 102,760,448  
 FC: [1x1x4096] memory: 4096 params: 4096\*4096 = 16,777,216  
 FC: [1x1x1000] memory: 1000 params: 4096\*1000 = 4,096,000  
 TOTAL memory: 24M \* 4 bytes ~ 96MB / image (only forward! ~\*2 for bwd)  
 TOTAL params: 138M parameters

Note:

Most memory is in early CONV

Most params are in late FC

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 53

April 17, 2024

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 54

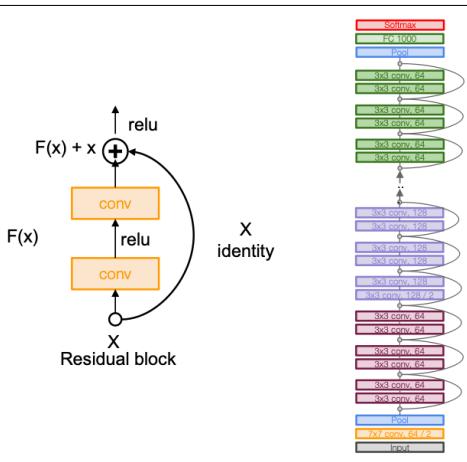
April 17, 2024

## Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



## Case Study: ResNet

[He et al., 2015]

Training ResNet in practice:

- Batch Normalization after every CONV layer
- Xavier initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

## Main takeaways

AlexNet showed that you can use CNNs to train Computer Vision models.  
VGG shows that bigger networks work better

ResNet showed us how to train extremely deep networks

- Limited only by GPU & memory!
- Showed diminishing returns as networks got bigger

After ResNet: CNNs were better than the human metric and focus shifted to other topics:

- Efficient Networks: **MobileNet**, **ShuffleNet**
- **Neural Architecture Search** can now automate architecture design

## Summary: CNN Architectures

- Many popular architectures are available in model zoos.
- ResNets are good defaults to use.  
True for > 8 years!
- Networks have gotten increasingly deep over time.
- Many other aspects of network architectures are also continuously being investigated and improved.

## Transfer learning

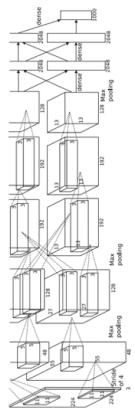
Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 81

April 17, 2024

You need a lot of data if you want to train/use CNNs?

## Transfer Learning with CNNs

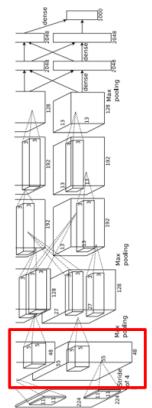


Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 83

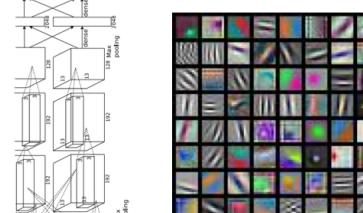
April 17, 2024

## Transfer Learning with CNNs



Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 84 April 17, 2024

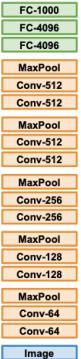


AlexNet:  
64 x 3 x 11 x 11

(More on this in Lecture 13)

## Transfer Learning with CNNs

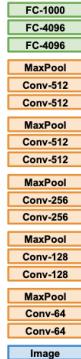
1. Train on Imagenet



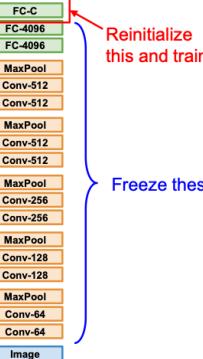
Donahue et al., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014  
Razavian et al., "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

## Transfer Learning with CNNs

1. Train on Imagenet



2. Small Dataset (C classes)



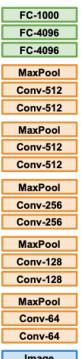
Donahue et al., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014  
Razavian et al., "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

Fei-Fei Li, Ehsan Adeli, Zane Durante

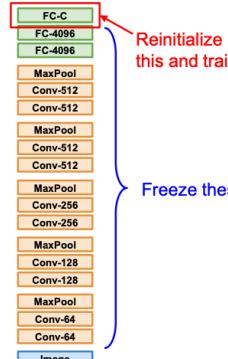
Lecture 6 - 86 April 17, 2024

## Transfer Learning with CNNs

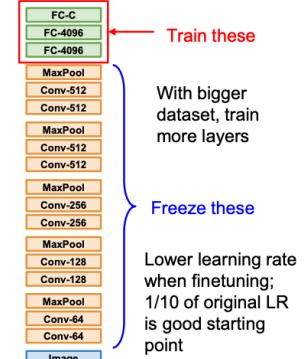
1. Train on Imagenet



2. Small Dataset (C classes)



3. Bigger dataset



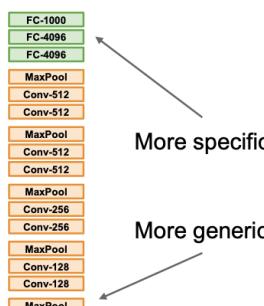
Donahue et al., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014  
Razavian et al., "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 89 April 17, 2024

Fei-Fei Li, Ehsan Adeli, Zane Durante

Lecture 6 - 90 April 17, 2024



	very similar dataset	very different dataset
very little data	?	?
quite a lot of data	?	?



More specific

More generic

	<b>very similar dataset</b>	<b>very different dataset</b>
<b>very little data</b>	Use Linear Classifier on top layer	?
<b>quite a lot of data</b>	Finetune a few layers	?



More specific

More generic

	<b>very similar dataset</b>	<b>very different dataset</b>
<b>very little data</b>	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
<b>quite a lot of data</b>	Finetune a few layers	Finetune a larger number of layers or start from scratch!