

# Convolutional Neural Network

Mingchen Gao

Slides adapted from Stanford CS231n

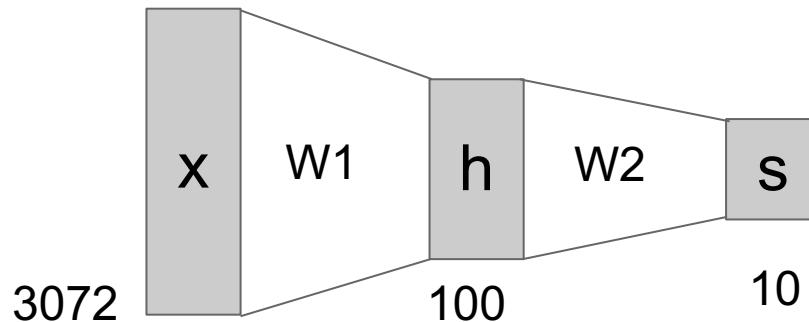
# Last time: Neural Networks

Linear score function:

$$f = Wx$$

2-layer Neural Network

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



# Next: Convolutional Neural Networks

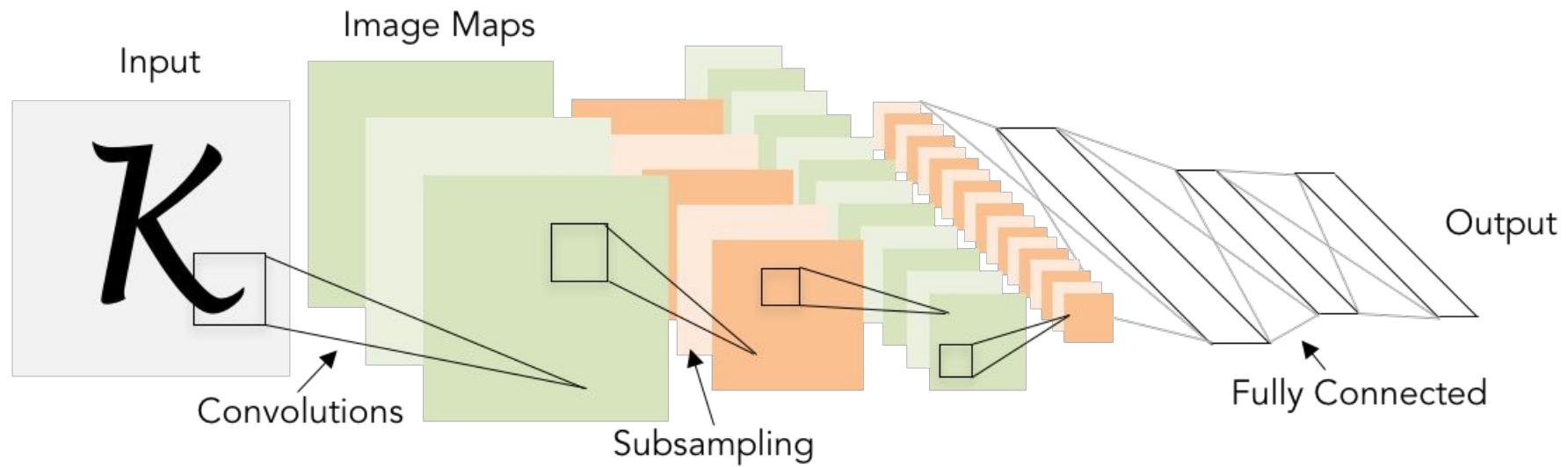
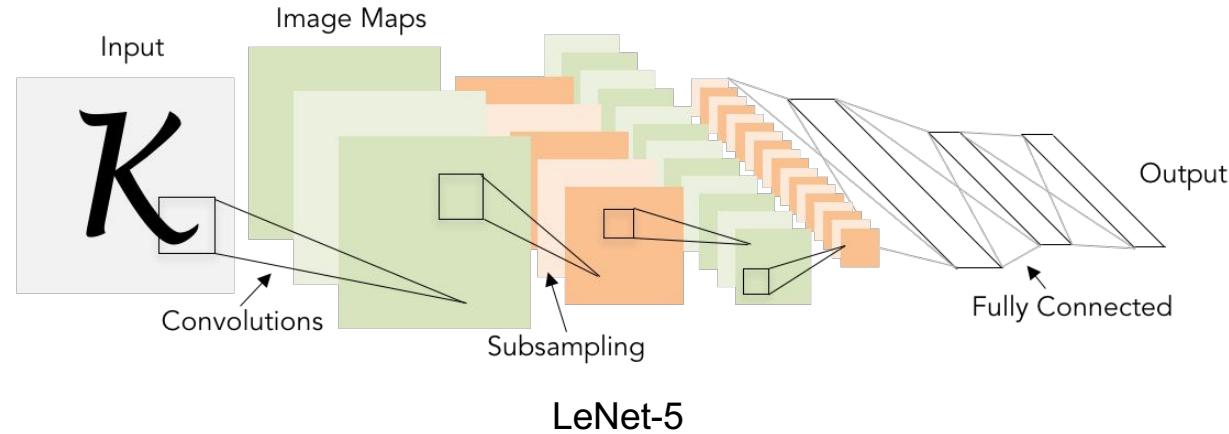


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

# A bit of history: **Gradient-based learning applied to document recognition** *[LeCun, Bottou, Bengio, Haffner 1998]*



# A bit of history:

## ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

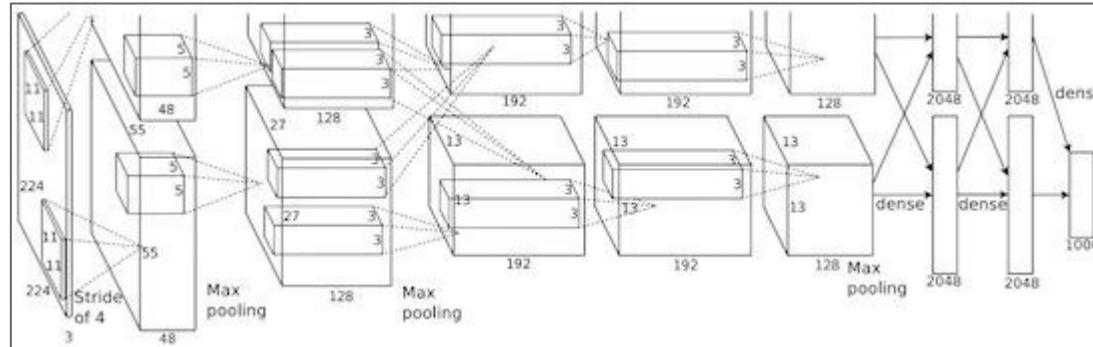
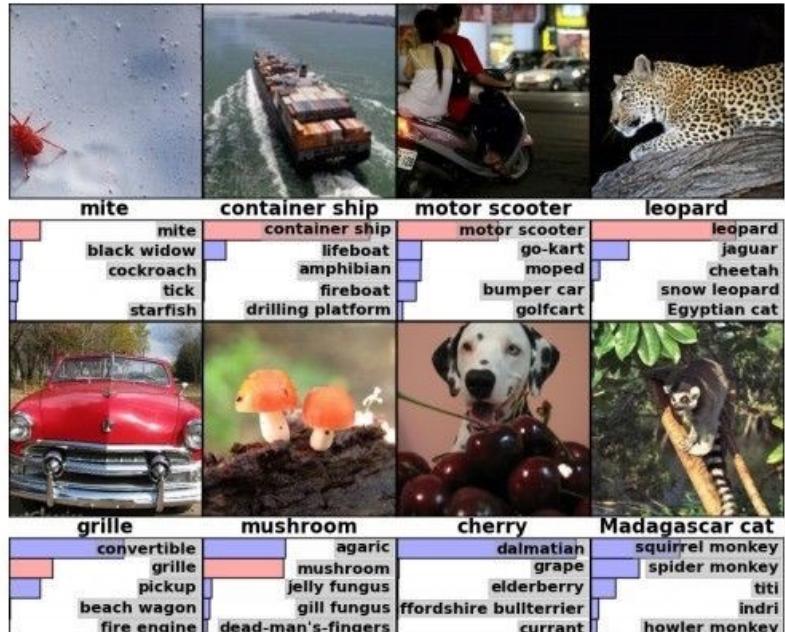


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

### “AlexNet”

# Fast-forward to today: ConvNets are everywhere

Classification



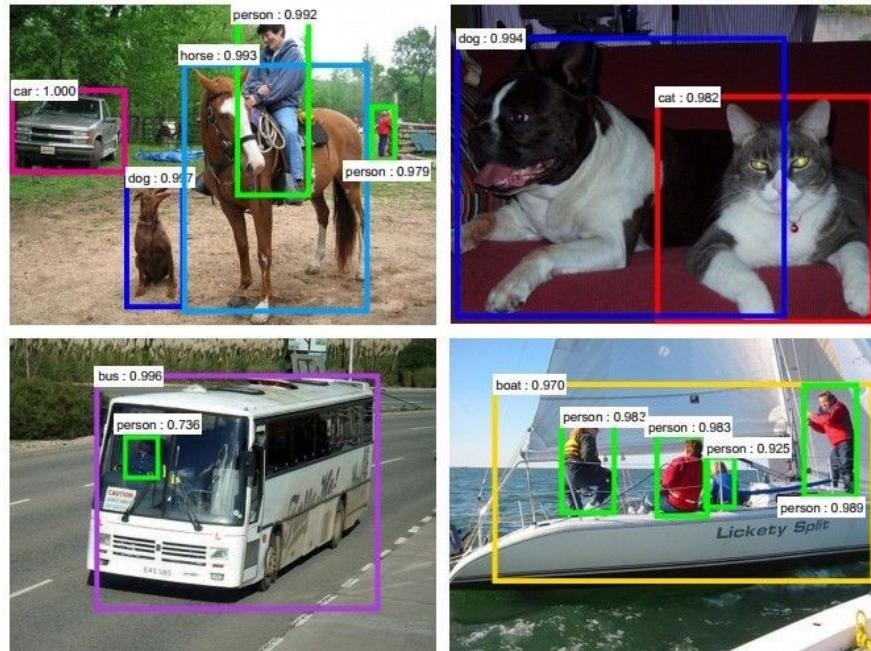
Retrieval



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

# Fast-forward to today: ConvNets are everywhere

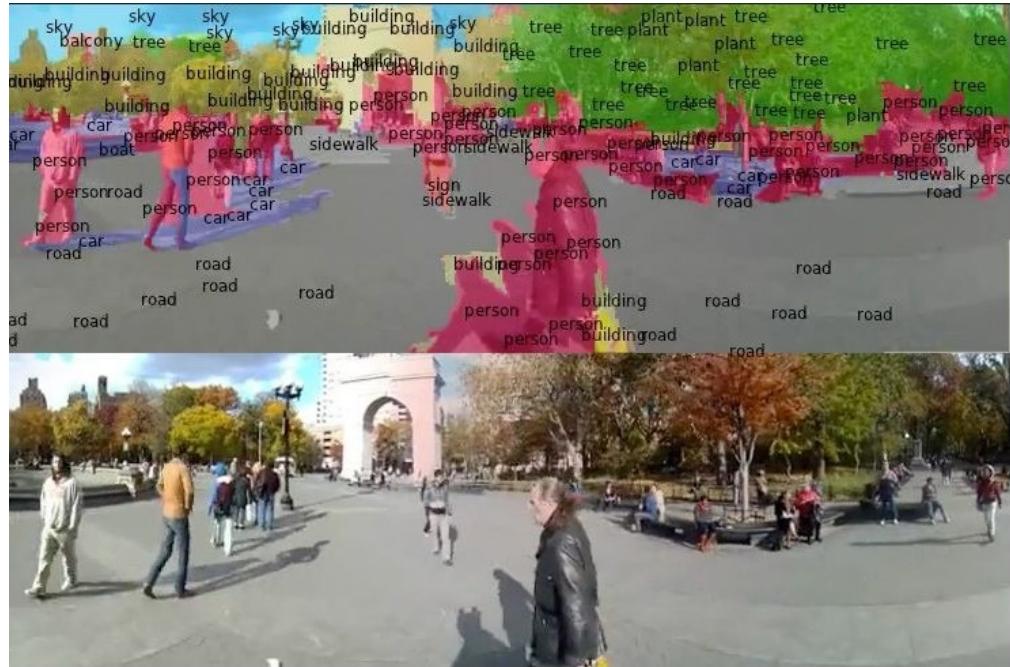
## Detection



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

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

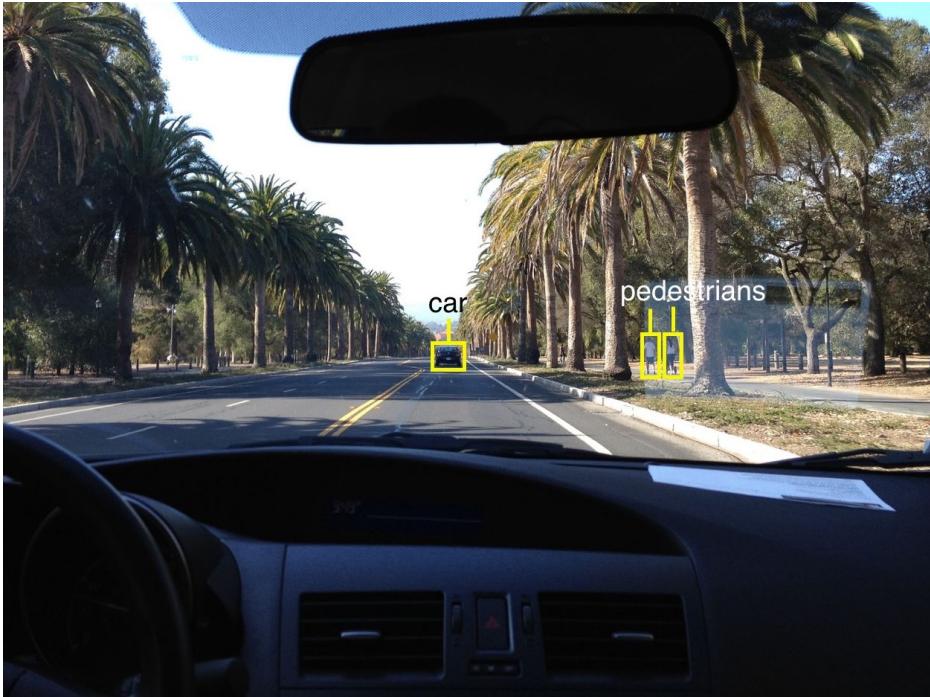
## Segmentation



Figures copyright Clement Farabet, 2012.  
Reproduced with permission.

[Farabet et al., 2012]

# Fast-forward to today: ConvNets are everywhere



self-driving cars

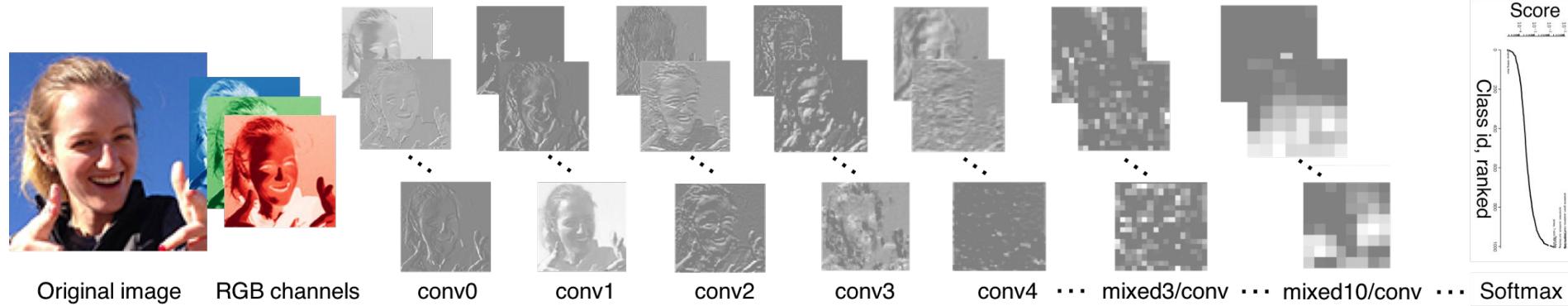
Photo by Lane McIntosh. Copyright CS231n 2017.



NVIDIA Tesla line

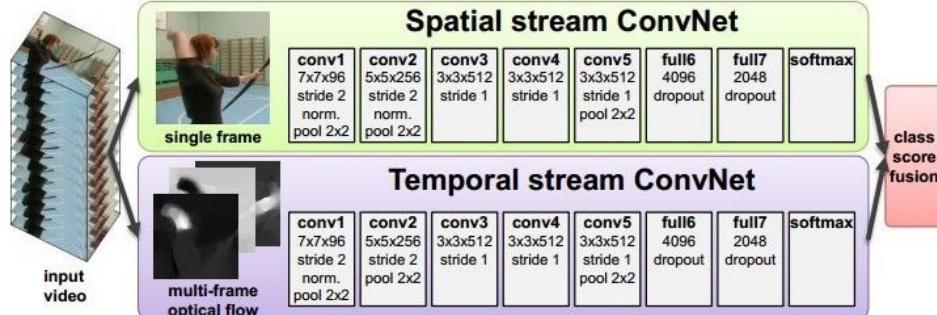
[This image](#) by GBPublic\_PR is  
licensed under [CCBY2.0](#)

# Fast-forward to today: ConvNets are everywhere



[Taigman et al. 2014]

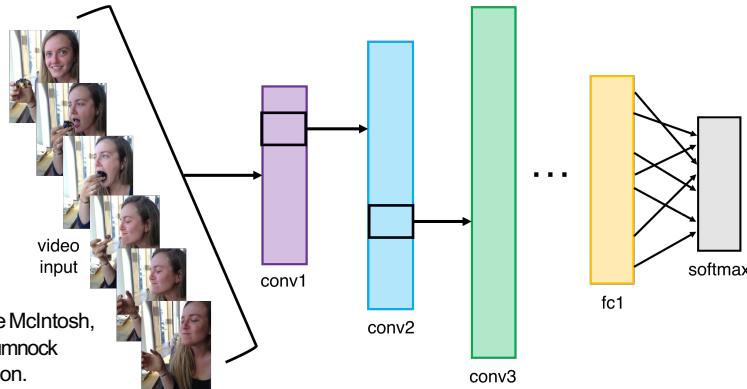
Activations of [inception-v3 architecture](#) [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014.  
Reproduced with permission.

Illustration by Lane McIntosh,  
photos of Katie Cumnock  
used with permission.

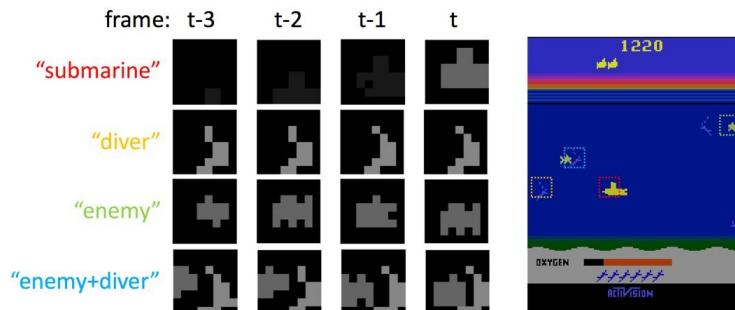


# Fast-forward to today: ConvNets are everywhere

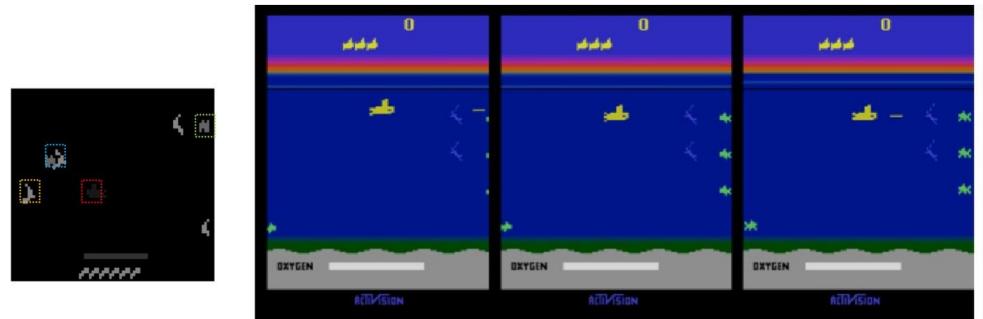


Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]

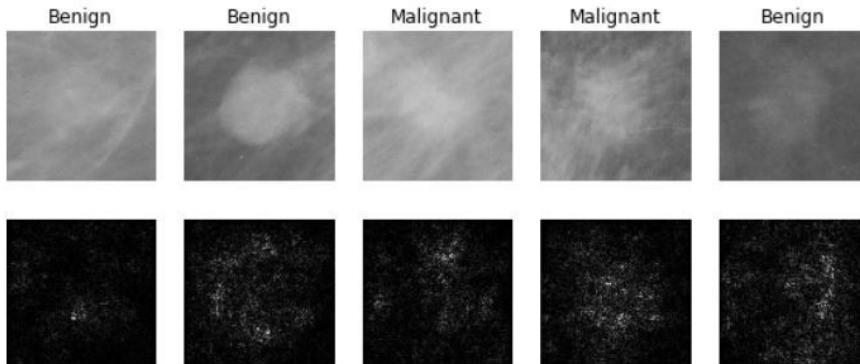


[Guo et al. 2014]



Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

# Fast-forward to today: ConvNets are everywhere



[Levy et al. 2016]

Figure copyright Levy et al. 2016.  
Reproduced with permission.



[Dieleman et al. 2014]

From left to right: [public domain by NASA](#), usage [permitted](#) by  
ESA/Hubble, [public domain by NASA](#), and [public domain](#).



Photos by Lane McIntosh.  
Copyright CS231n 2017.

[Sermanet et al. 2011]  
[Ciresan et al.]

[This image](#) by Christin Khan is in the public domain and originally came from the U.S. NOAA.



*Whale recognition, Kaggle Challenge*

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.



*Mnih and Hinton, 2010*

No errors



*A white teddy bear sitting in the grass*



*A man riding a wave on top of a surfboard*

Minor errors



*A man in a baseball uniform throwing a ball*



*A cat sitting on a suitcase on the floor*

Somewhat related



*A woman is holding a cat in her hand*



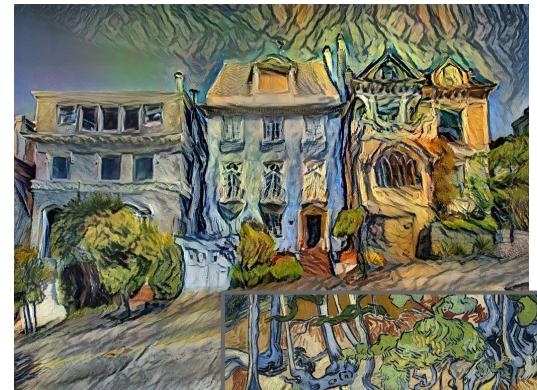
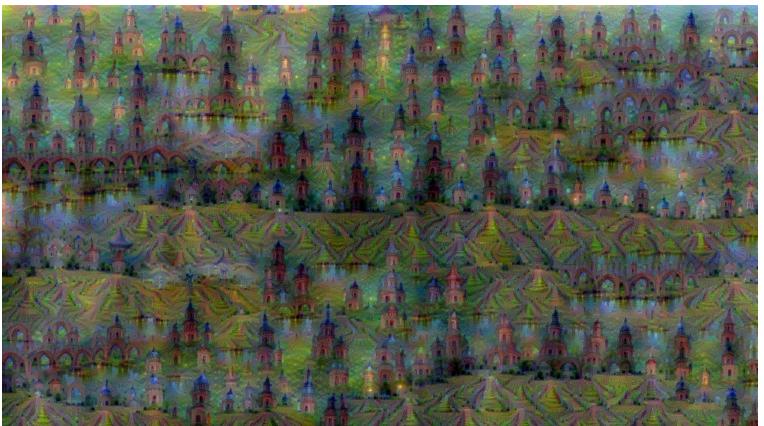
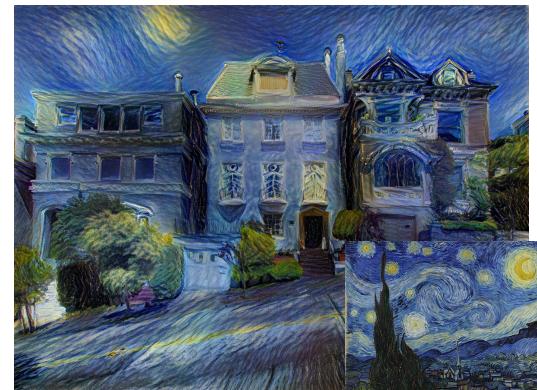
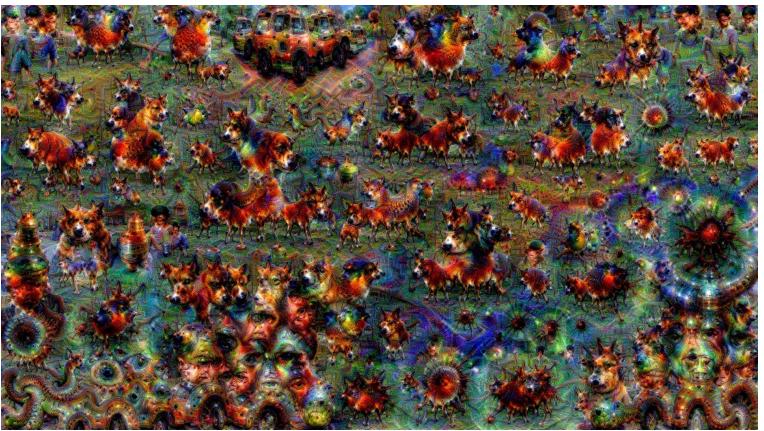
*A woman standing on a beach holding a surfboard*

# Image Captioning

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

All images are CC0 Public domain:  
<https://pixabay.com/en/luggage-antique-cat-1643010/>  
<https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/>  
<https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/>  
<https://pixabay.com/en/woman-female-model-portrait-adult-983967/>  
<https://pixabay.com/en/handstand-lake-meditation-496008/>  
<https://pixabay.com/en/baseball-player-shortstop-infield-1045263/>

Captions generated by Justin Johnson using [Neuraltalk2](#)



Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a [blog post](#) by Google Research.

[Original image](#) is CC0 public domain  
[Starry Night](#) and [Tree Roots](#) by Van Gogh are in the public domain  
[Bokeh image](#) is in the public domain  
Stylized images copyright Justin Johnson, 2017;  
reproduced with permission

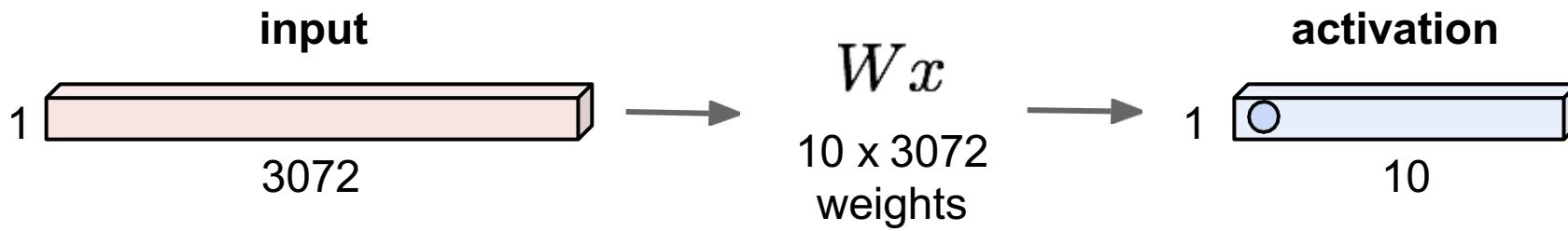
Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR2016  
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR2017

# Convolutional Neural Networks

(First without the brain stuff)

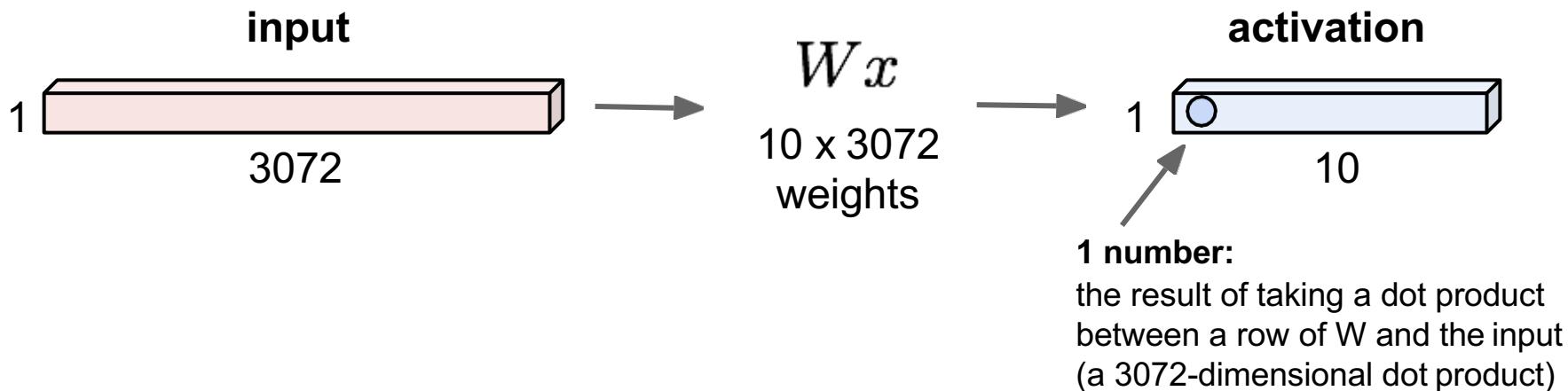
# Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



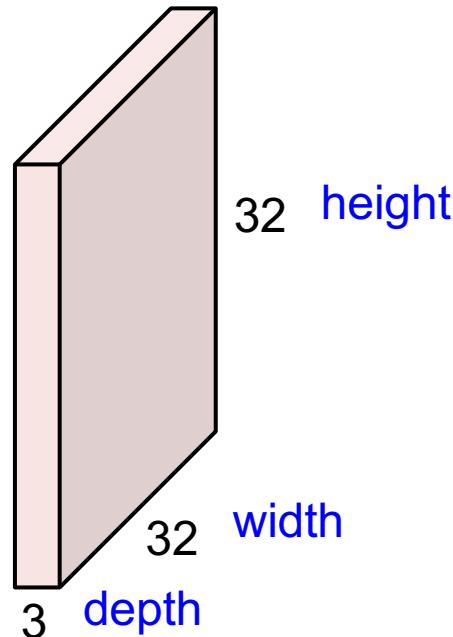
# Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



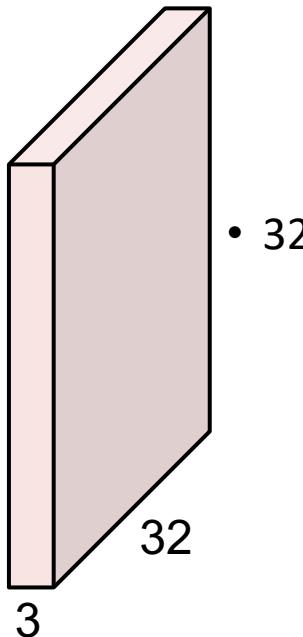
# Convolution Layer

32x32x3 image -> preserve spatial structure

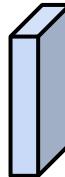


# Convolution Layer

- 32x32x3 image



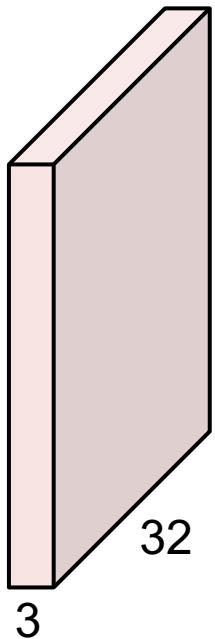
- 5x5x3 filter



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

# Convolution Layer

- 32x32x3 image



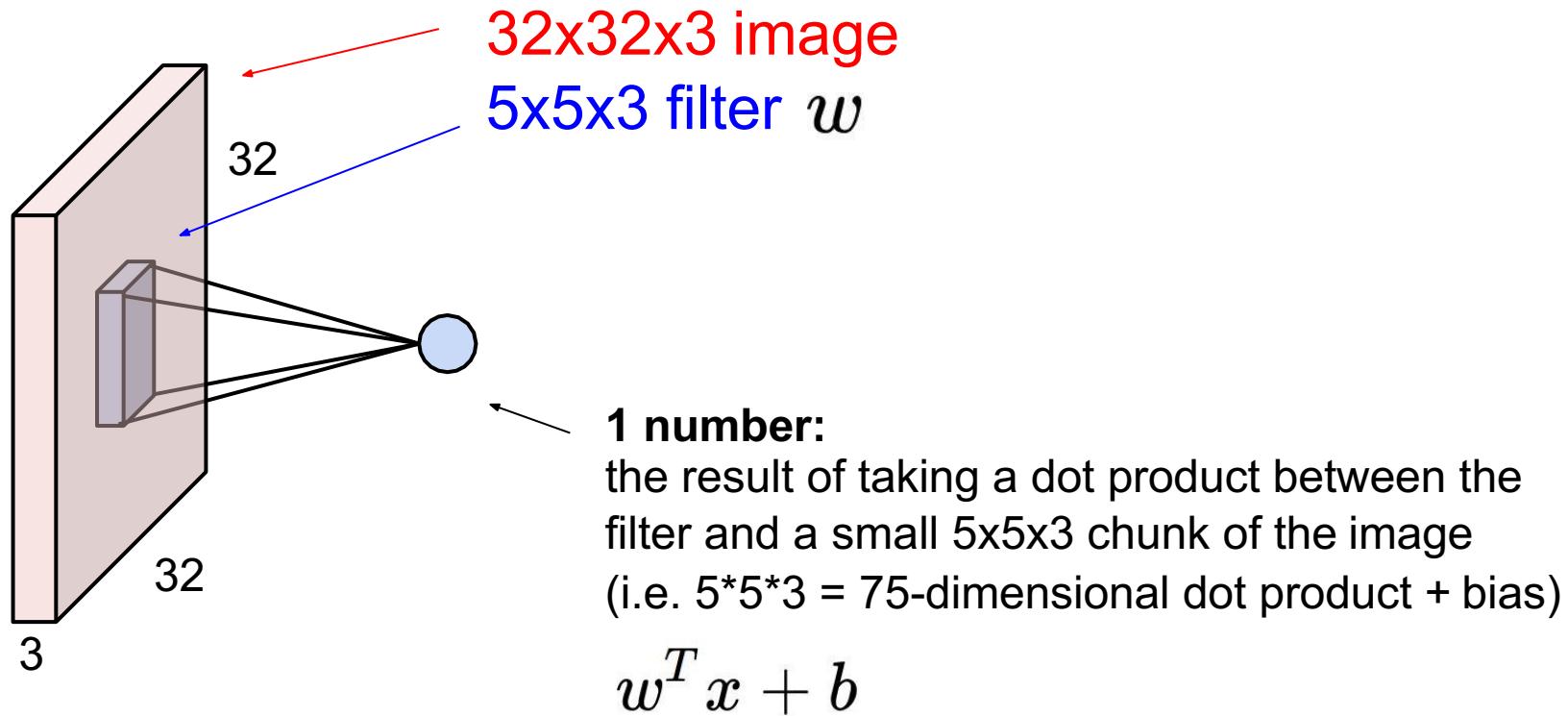
Filters always extend the full depth of the input volume

- 5x5x3 filter

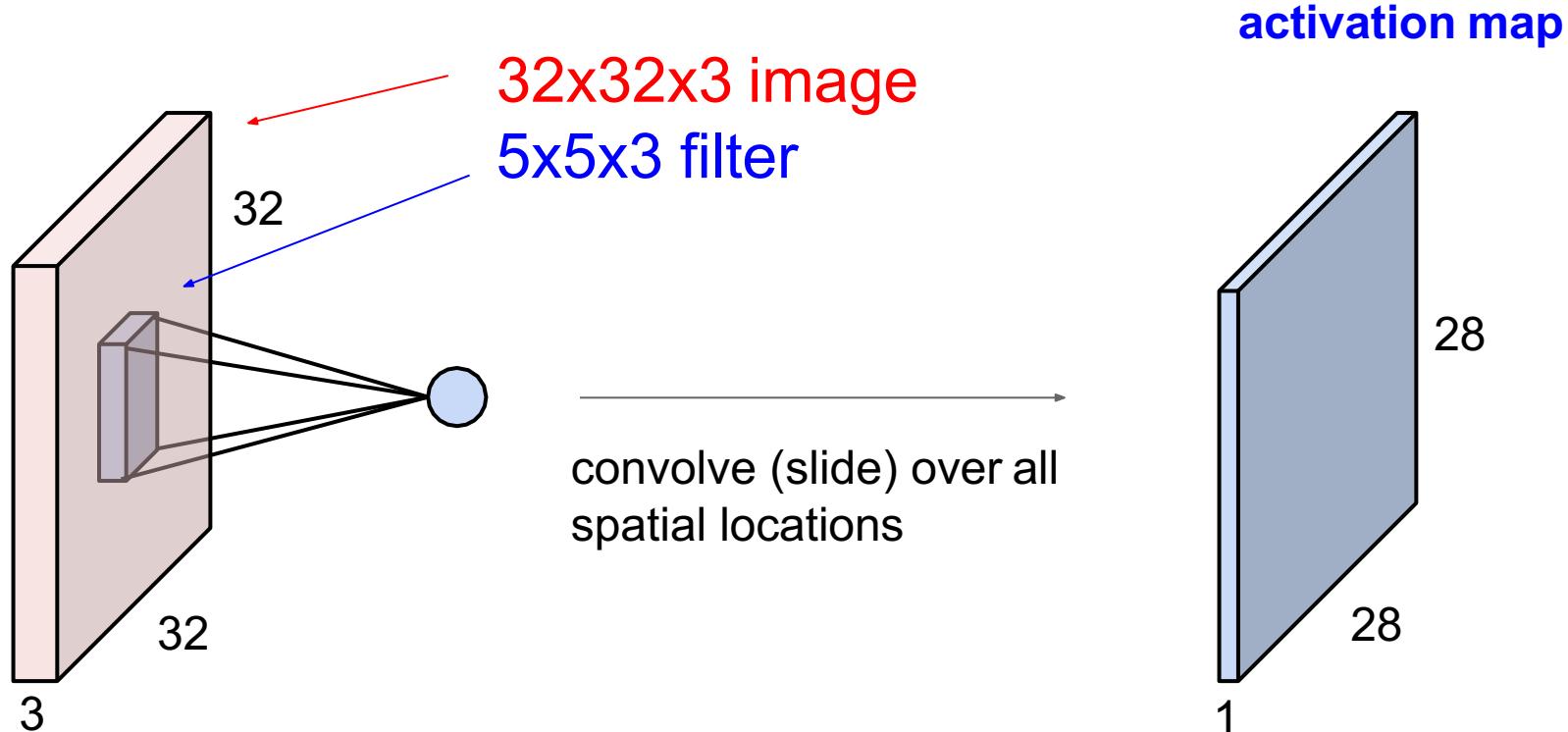


- **Convolve** the filter with the image
- i.e. “slide over the image spatially, computing dot products”

# Convolution Layer

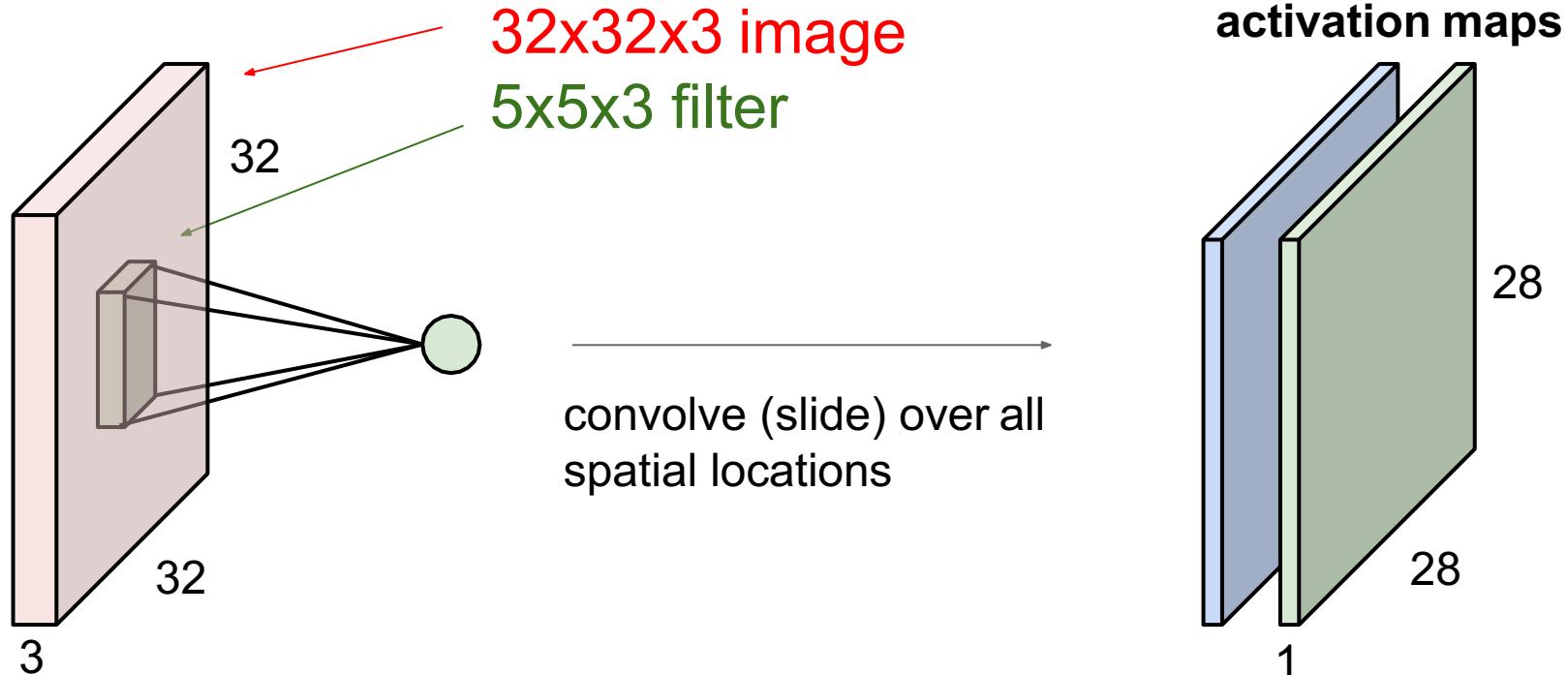


# Convolution Layer

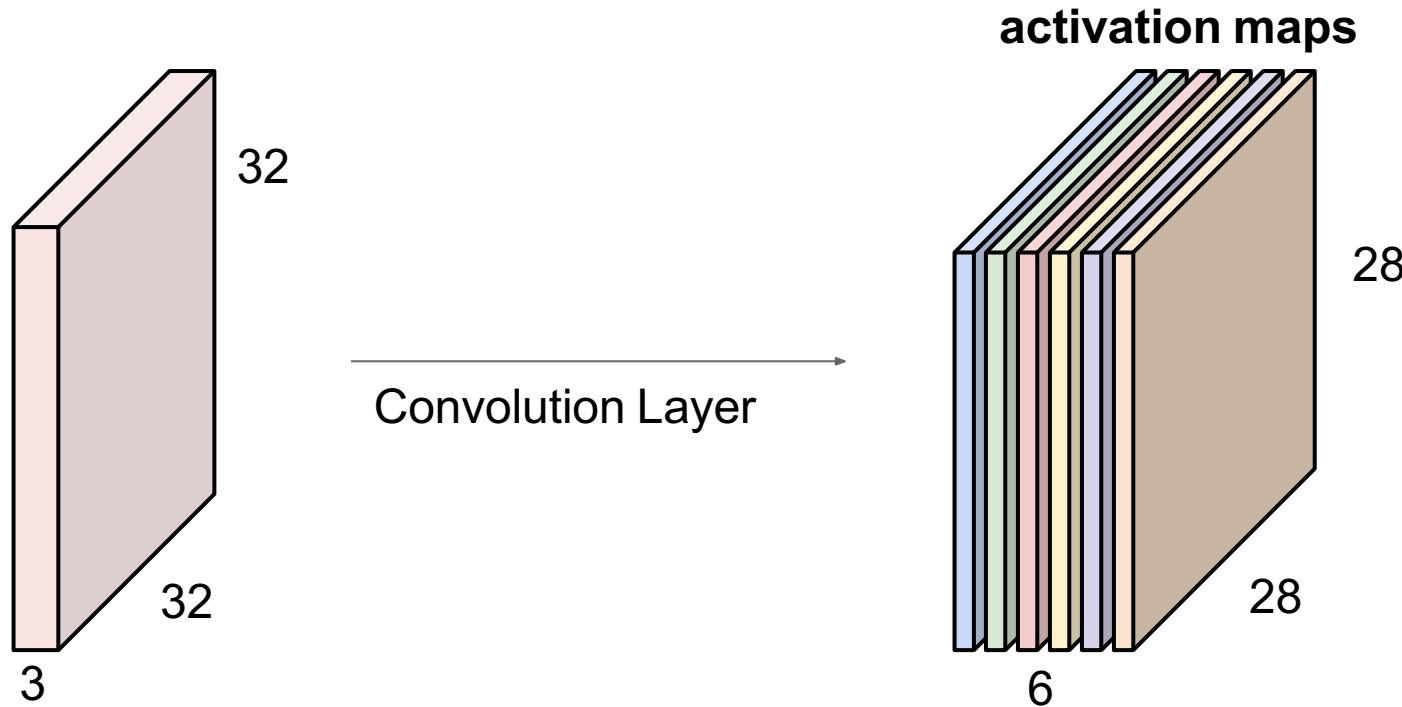


# Convolution Layer

consider a second, green filter

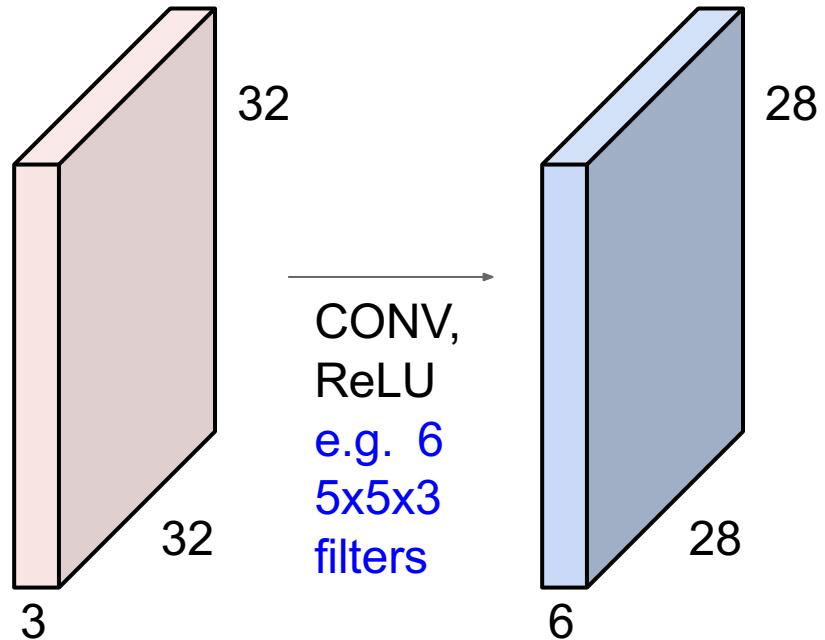


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

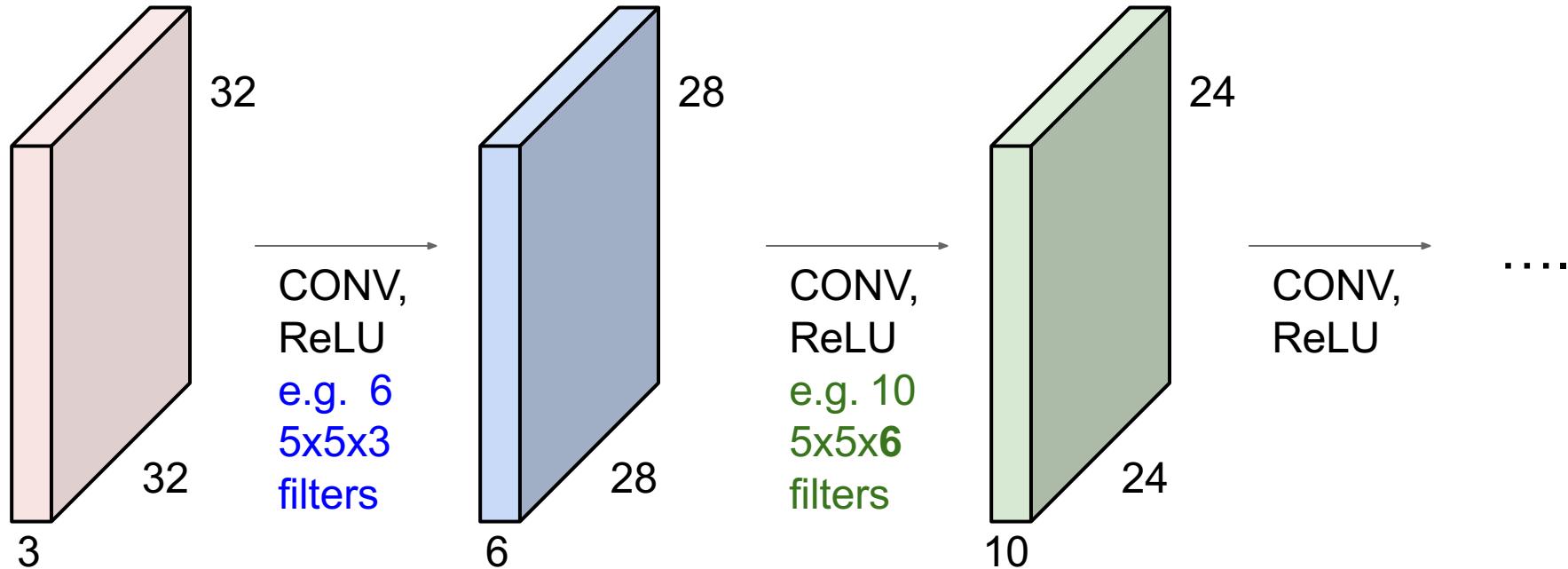


We stack these up to get a “new image” of size  $28 \times 28 \times 6$ !

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



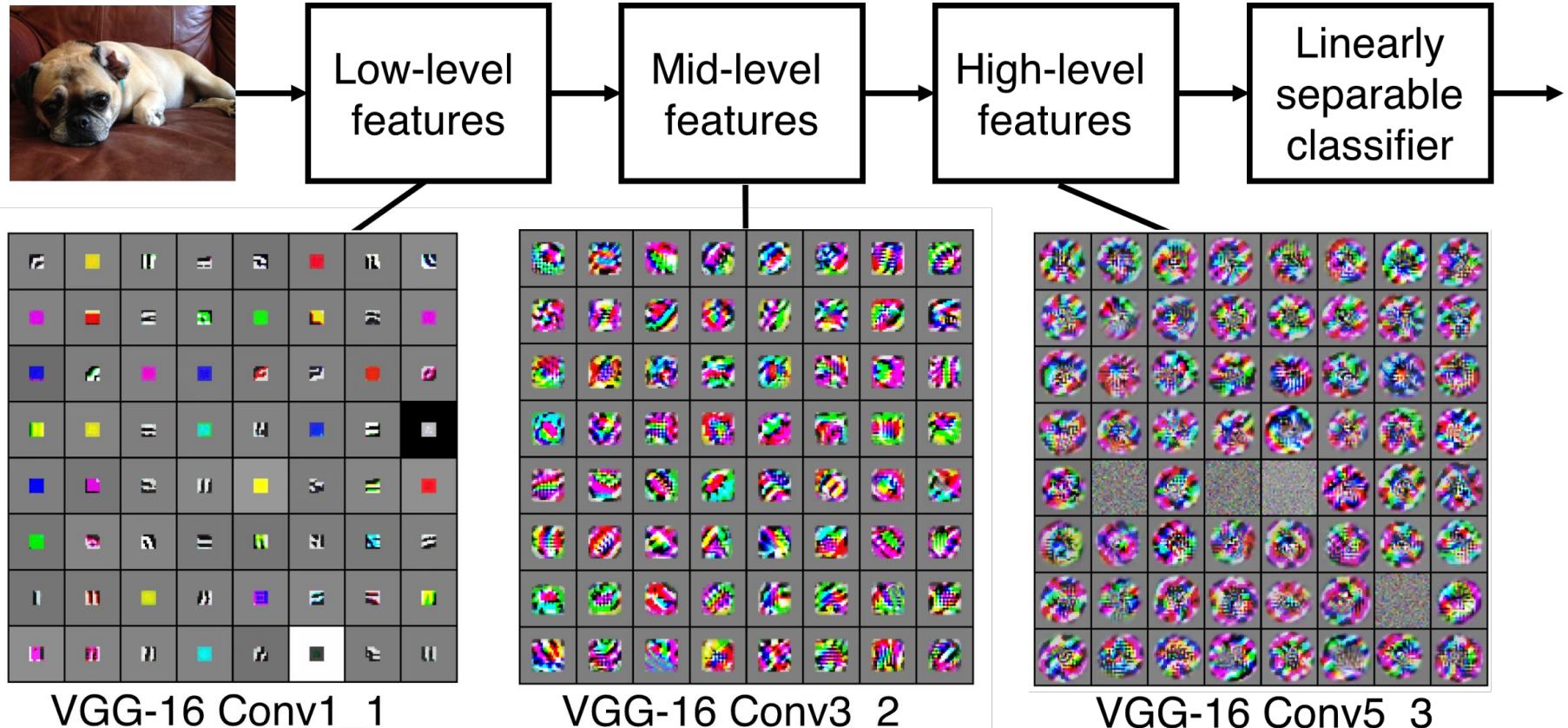
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



## Preview

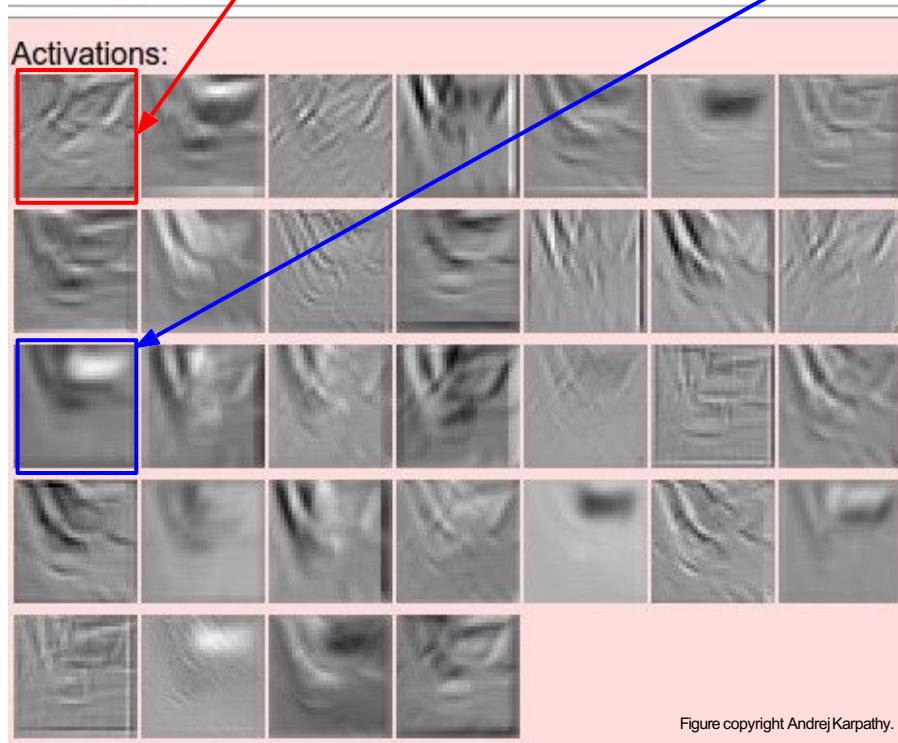
[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].





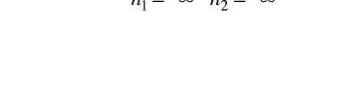
one filter =>  
one activation map



example 5x5 filters  
(32 total)

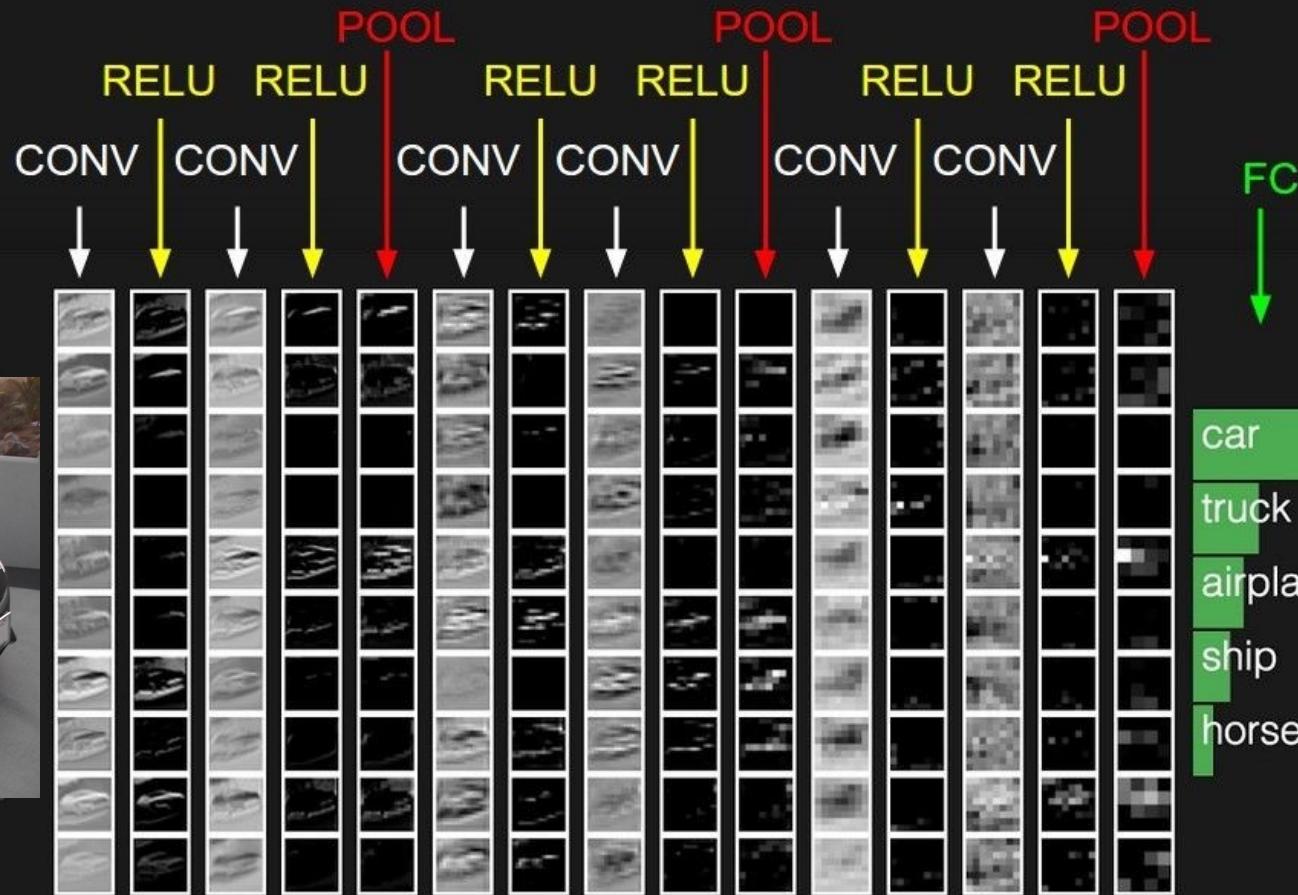
We call the layer convolutional  
because it is related to convolution  
of two signals:

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

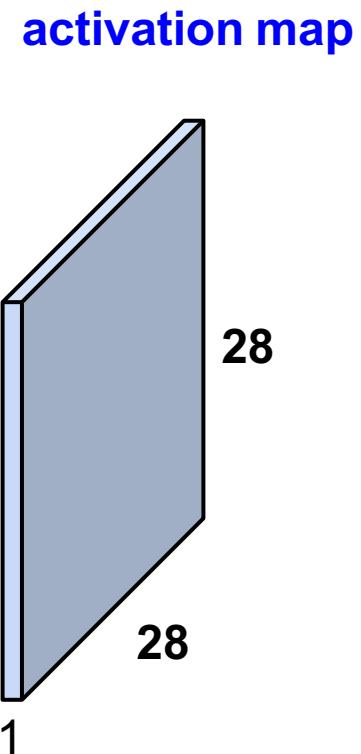
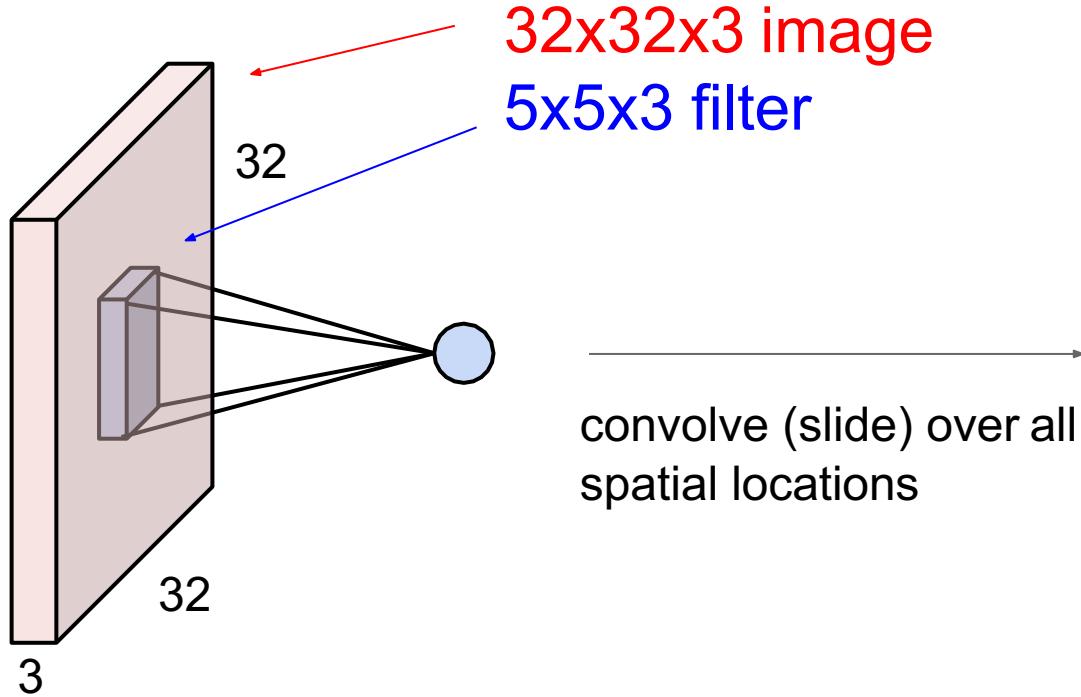


elementwise multiplication and sum of  
a filter and the signal (image)

preview:

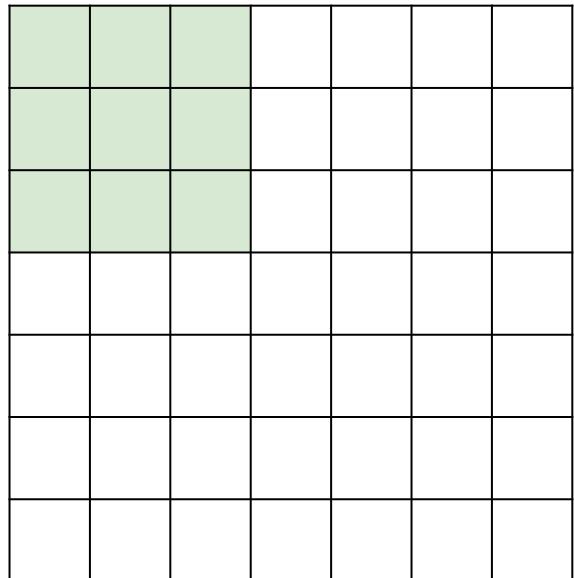


A closer look at spatial dimensions:



A closer look at spatial dimensions:

7



7x7 input (spatially)  
assume 3x3 filter

7

A closer look at spatial dimensions:

7


7x7 input (spatially)  
assume 3x3 filter

7

A closer look at spatial dimensions:

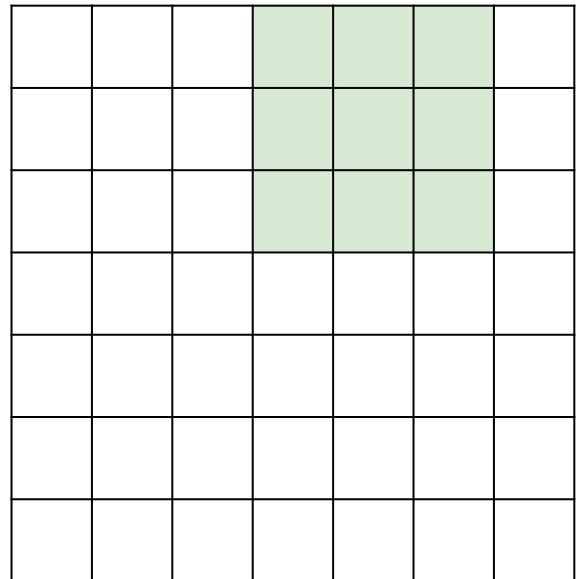
7


7x7 input (spatially)  
assume 3x3 filter

7

A closer look at spatial dimensions:

7

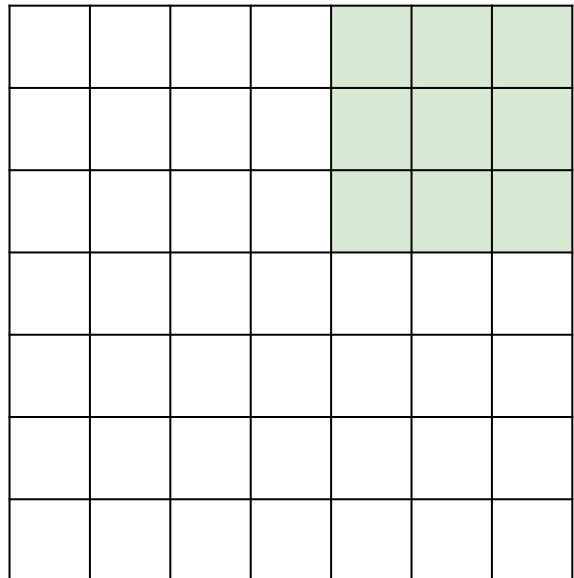


7x7 input (spatially)  
assume 3x3 filter

7

A closer look at spatial dimensions:

7

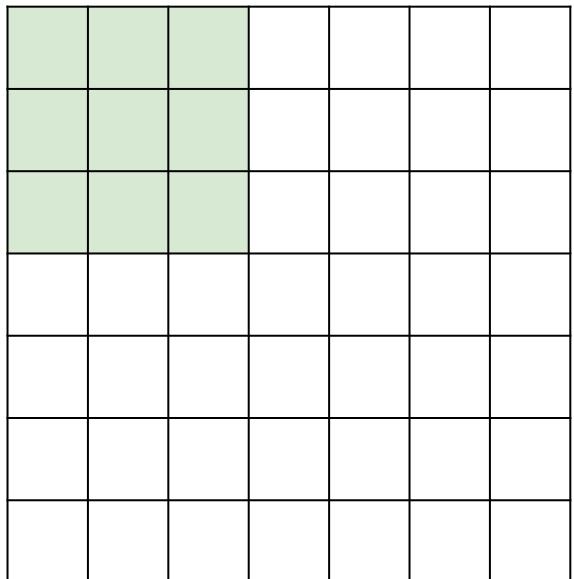


7x7 input (spatially)  
assume 3x3 filter

**=> 5x5 output**

A closer look at spatial dimensions:

7



7

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

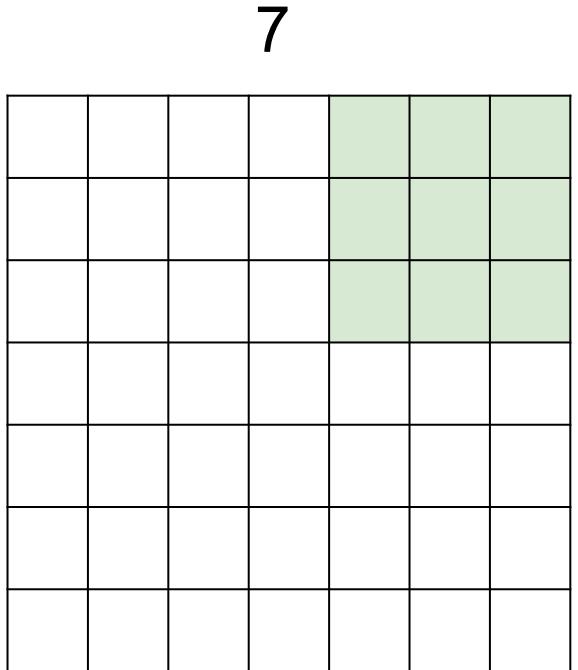
A closer look at spatial dimensions:

7


7

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

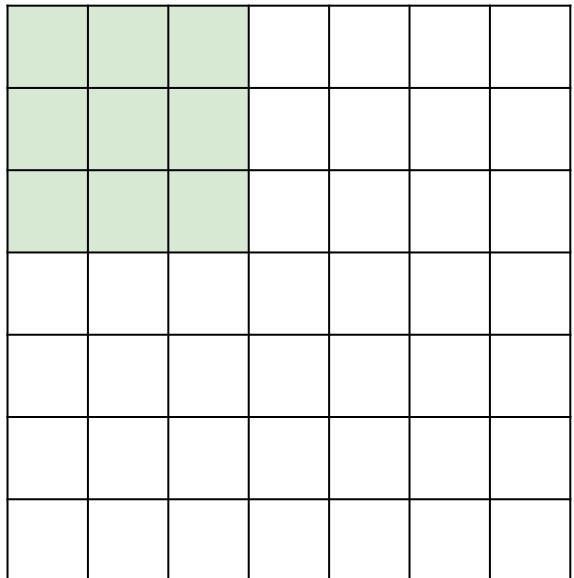
A closer look at spatial dimensions:



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

A closer look at spatial dimensions:

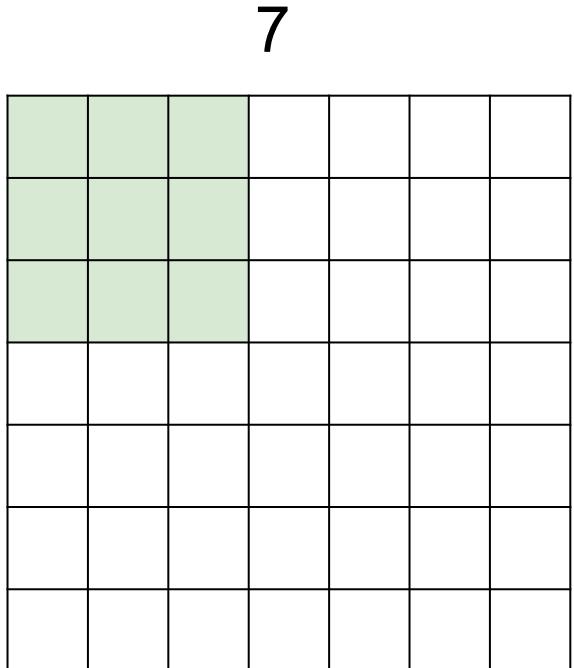
7



7

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

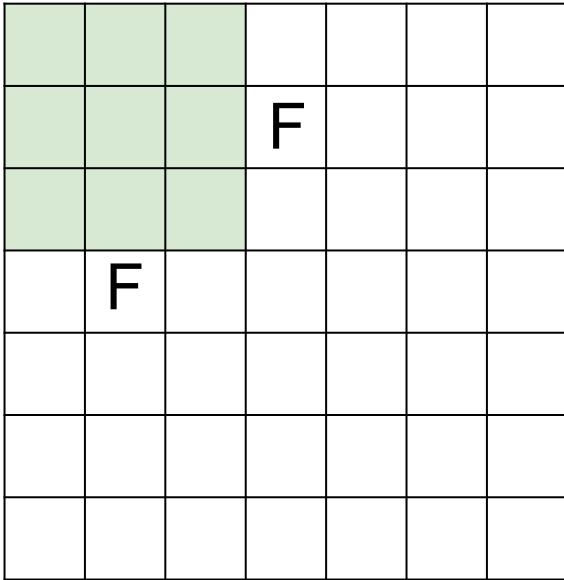
A closer look at spatial dimensions:



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

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

N



N

Output size:

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

e.g.  $N = 7$ ,  $F = 3$ :

$$\text{stride } 1 \Rightarrow (7 - 3)/1 + 1 = 5$$

$$\text{stride } 2 \Rightarrow (7 - 3)/2 + 1 = 3$$

$$\text{stride } 3 \Rightarrow (7 - 3)/3 + 1 = 2.33 :\backslash$$

# In practice: Common to zero pad the border

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?**

(recall:)

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

# In practice: Common to zero pad the border

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 practice: Common to zero pad the border

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)

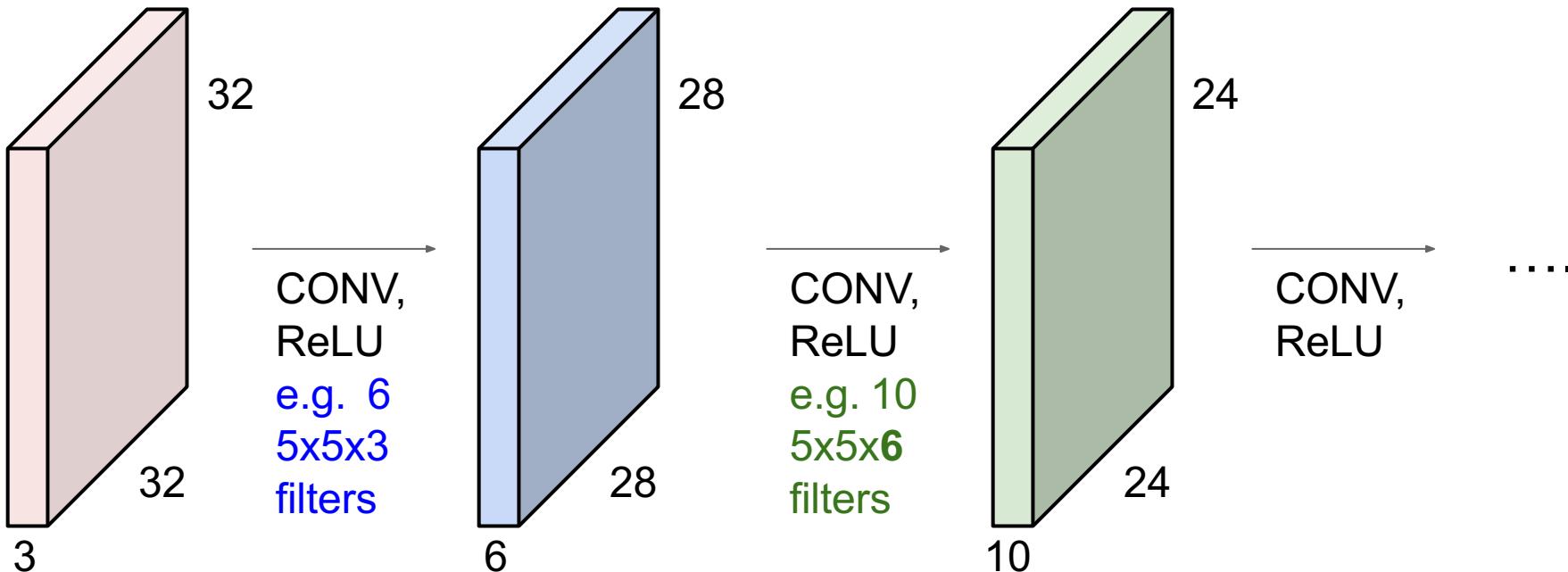
e.g.  $F = 3 \Rightarrow$  zero pad with 1

$F = 5 \Rightarrow$  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.

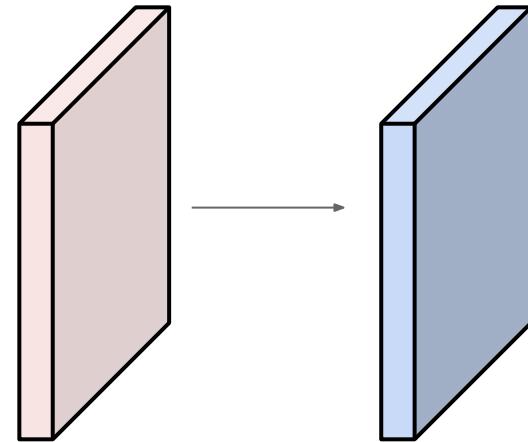


Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size: ?



Examples time:

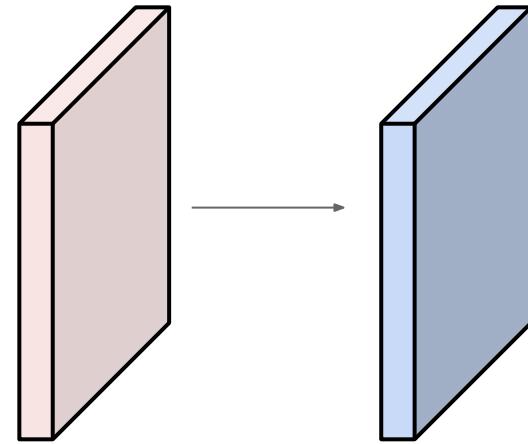
Input volume: **32x32x3**

10 **5x5** filters with stride 1, pad **2**

Output volume size:

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

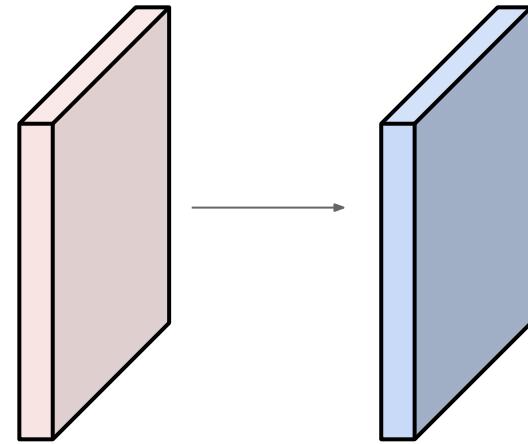
**32x32x10**



Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

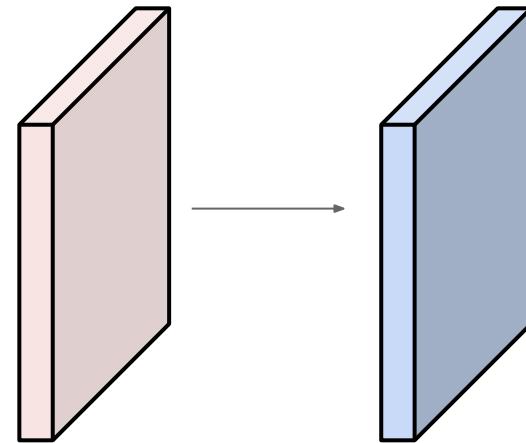


Number of parameters in this layer?

Examples time:

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)  
=>  $76*10 = 760$

**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters  $K$ ,
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
  - the amount of zero padding  $P$ .
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$  (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces  $F \cdot F \cdot D_1$  weights per filter, for a total of  $(F \cdot F \cdot D_1) \cdot K$  weights and  $K$  biases.
- In the output volume, the  $d$ -th depth slice (of size  $W_2 \times H_2$ ) is the result of performing a valid convolution of the  $d$ -th filter over the input volume with a stride of  $S$ , and then offset by  $d$ -th bias.

## Common settings:

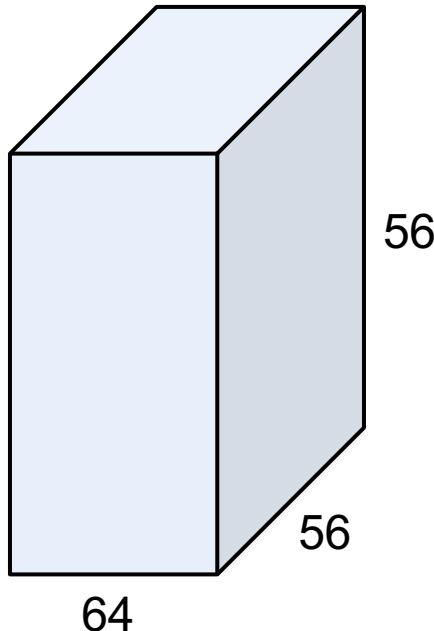
**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters  $K$ ,
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
  - the amount of zero padding  $P$ .
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$  (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces  $F \cdot F \cdot D_1$  weights per filter, for a total of  $(F \cdot F \cdot D_1) \cdot K$  weights and  $K$  biases.
- In the output volume, the  $d$ -th depth slice (of size  $W_2 \times H_2$ ) is the result of performing a valid convolution of the  $d$ -th filter over the input volume with a stride of  $S$ , and then offset by  $d$ -th bias.

$$K = (\text{powers of 2, e.g. } 32, 64, 128, 512)$$

- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ? \text{ (whatever fits)}$
- $F = 1, S = 1, P = 0$

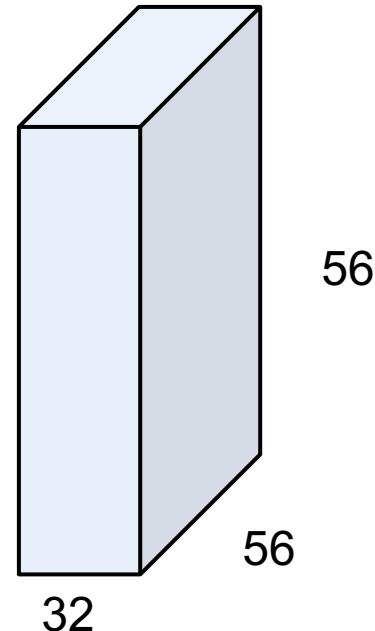
(btw, 1x1 convolution layers make perfect sense)



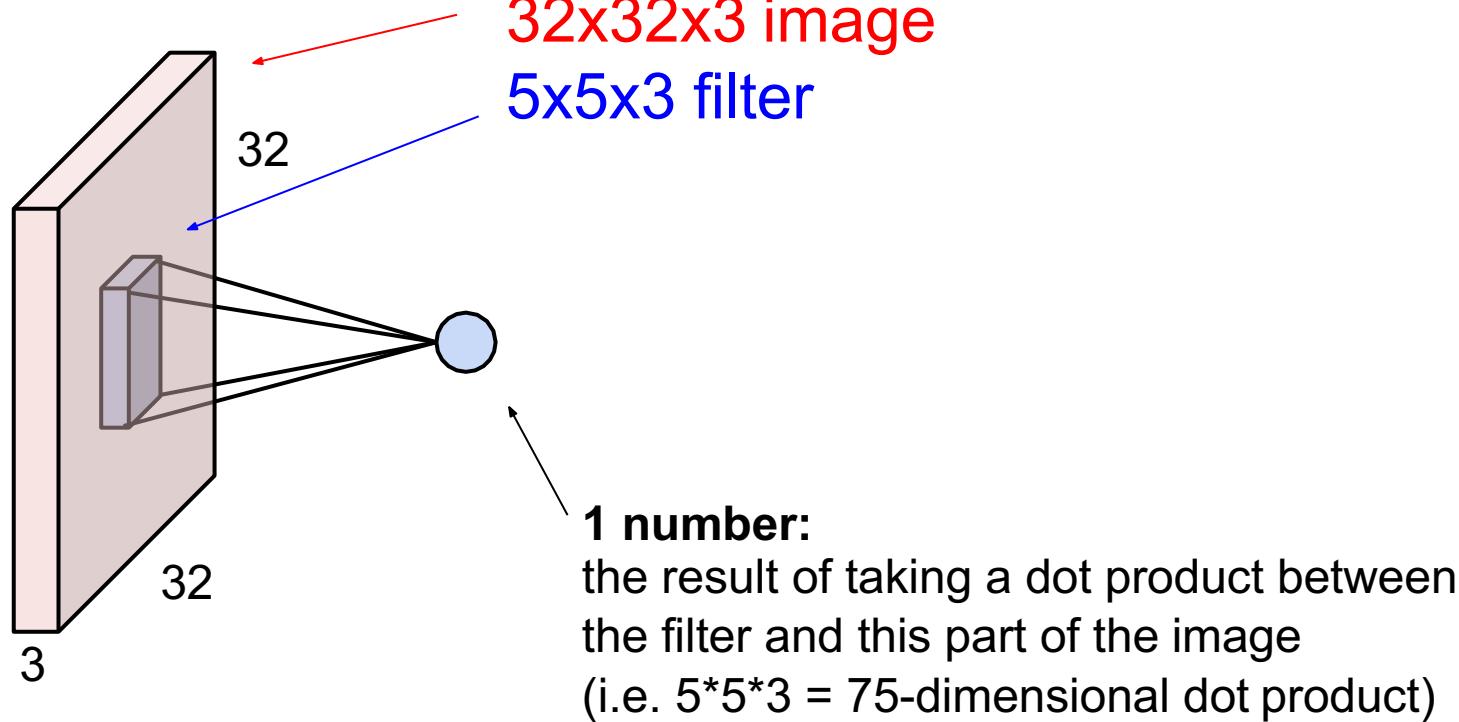
1x1 CONV  
with 32 filters

---

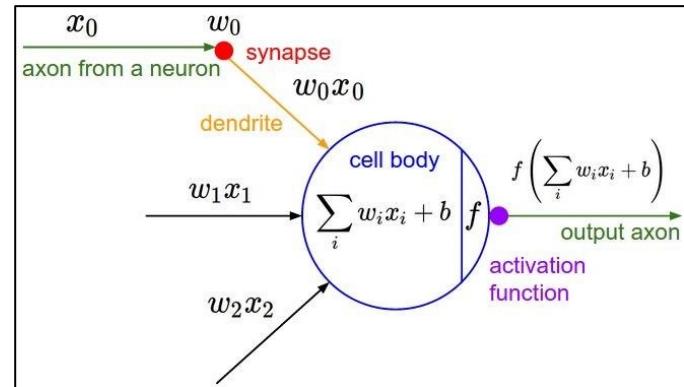
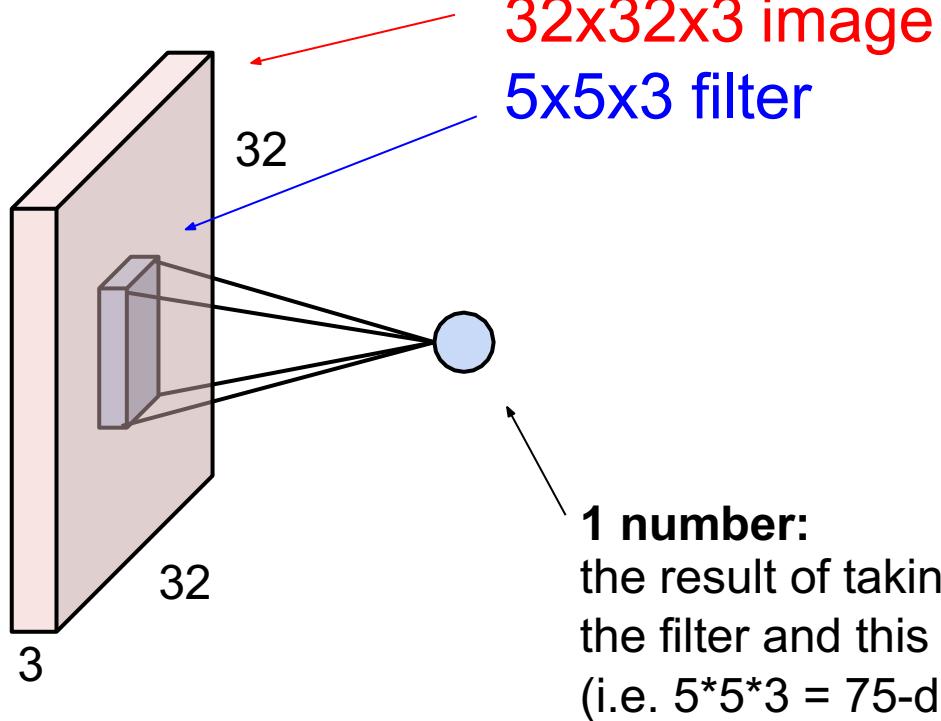
(each filter has size  
1x1x64, and performs a  
64-dimensional dot  
product)



# The brain/neuron view of CONV Layer

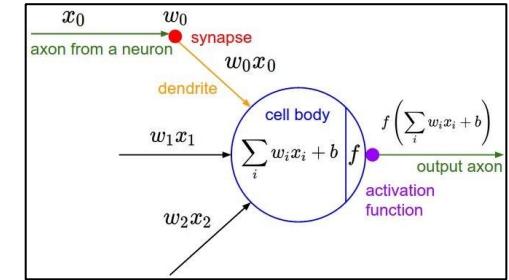
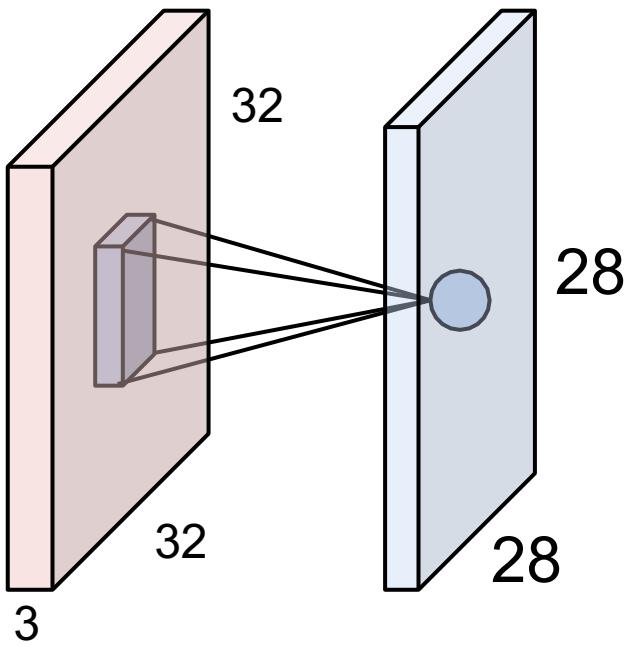


# The brain/neuron view of CONV Layer



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

# The brain/neuron view of CONV Layer

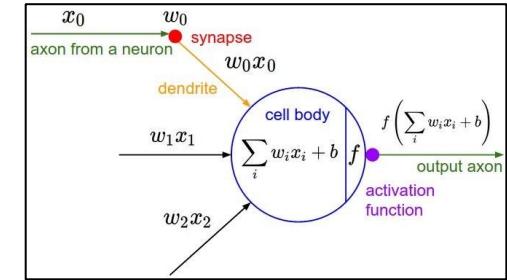
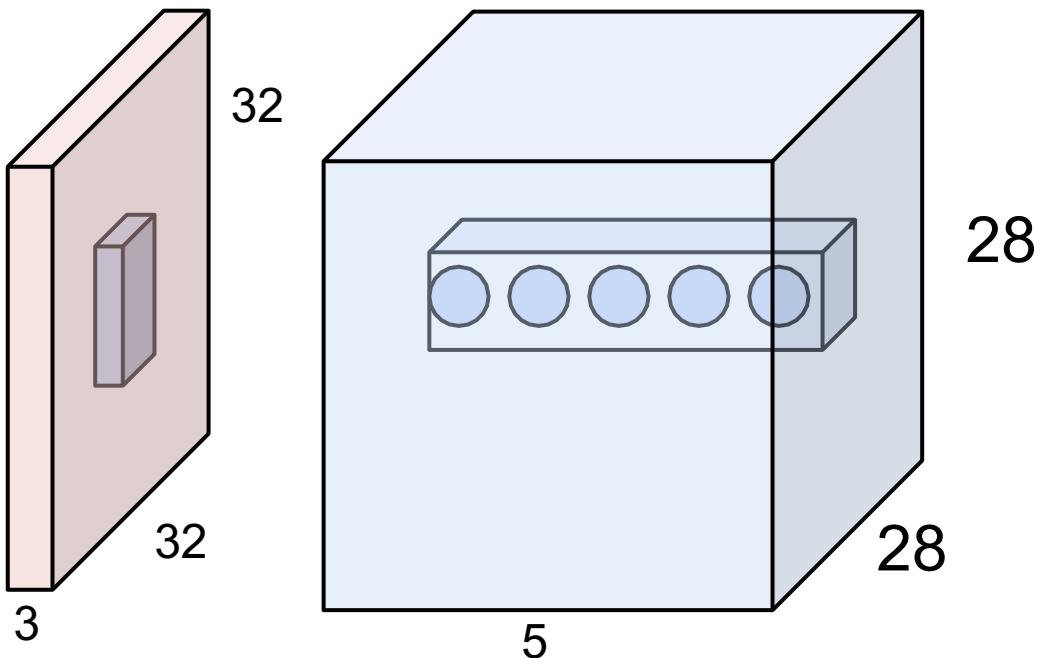


An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

“5x5 filter” -> “5x5 receptive field for each neuron”

# The brain/neuron view of CONV Layer



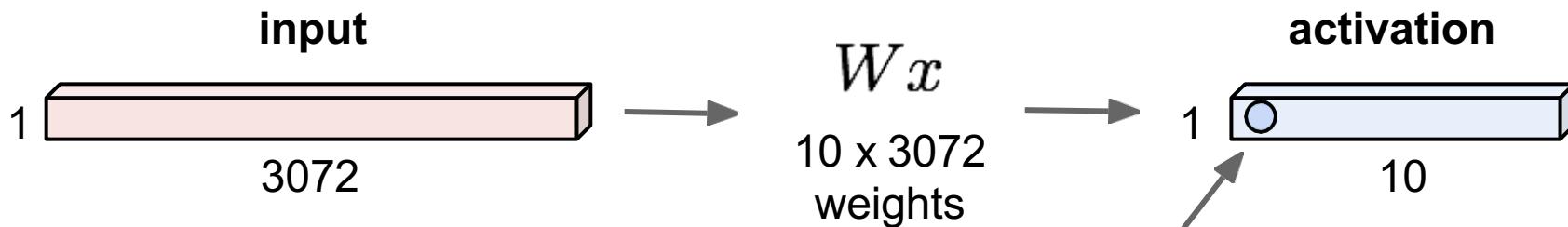
E.g. with 5 filters,  
CONV layer consists of  
neurons arranged in a 3D grid  
( $28 \times 28 \times 5$ )

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

# Reminder: Fully Connected Layer

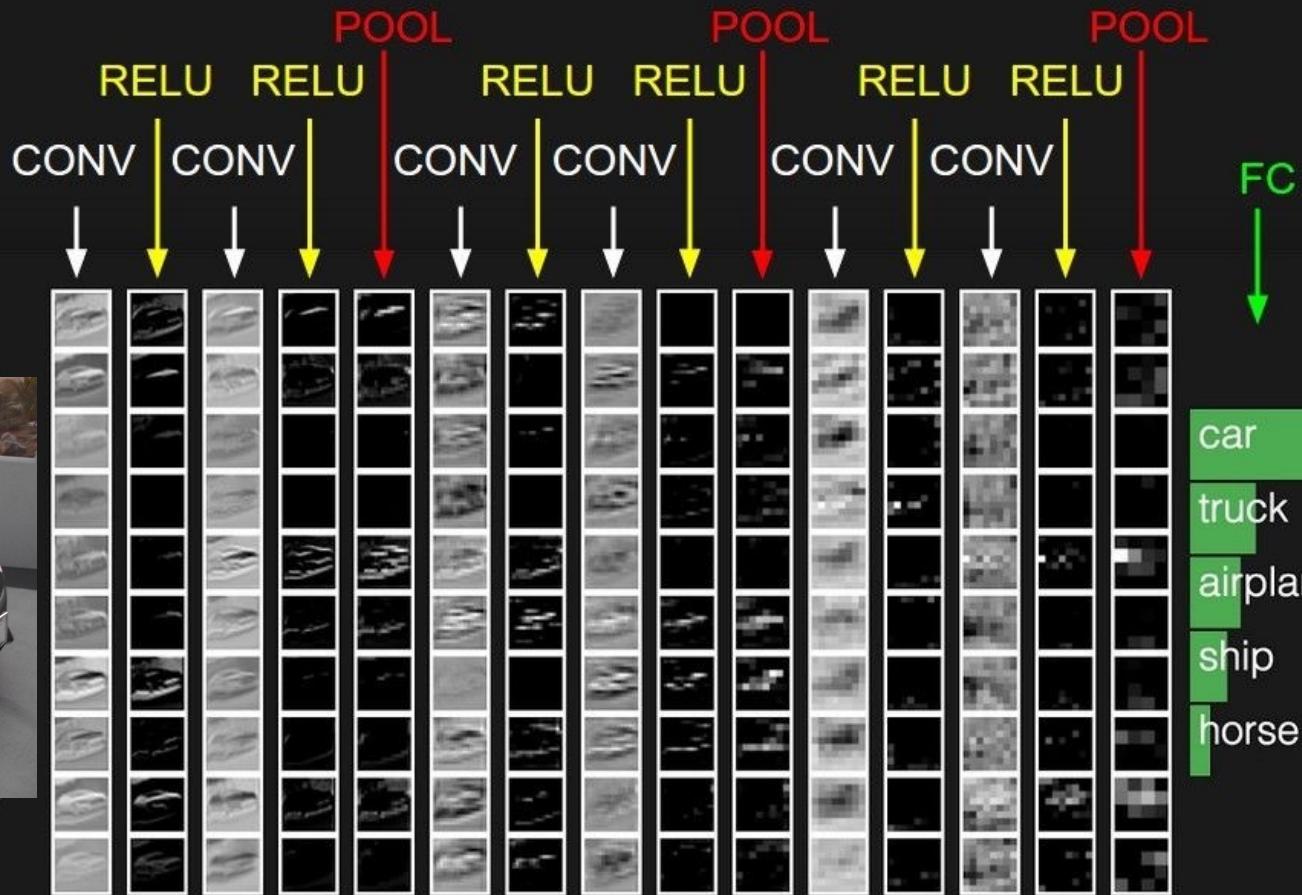
32x32x3 image -> stretch to 3072 x 1

Each neuron  
looks at the full  
input volume



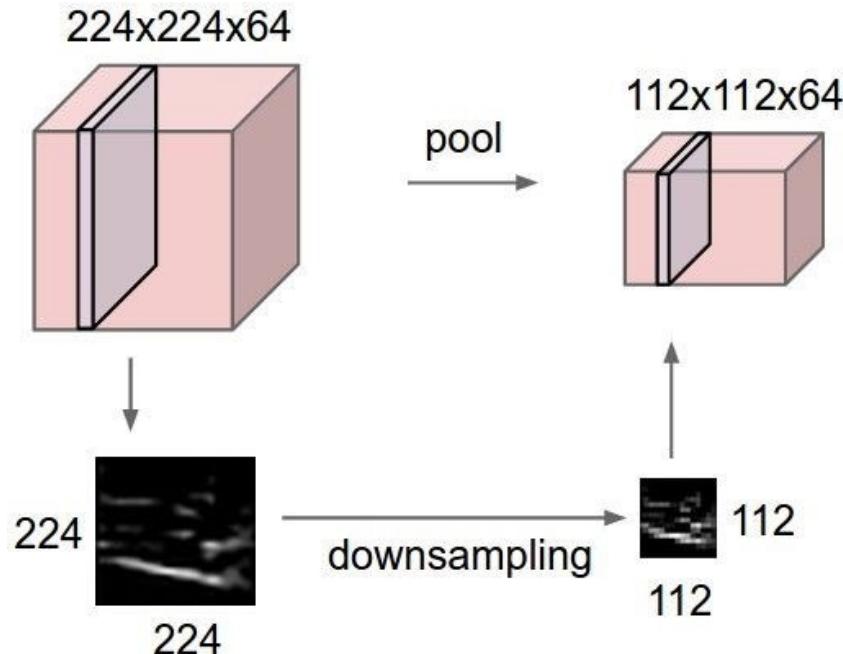
**1 number:**  
the result of taking a dot product  
between a row of  $W$  and the input  
(a 3072-dimensional dot product)

two more layers to go: POOL/FC



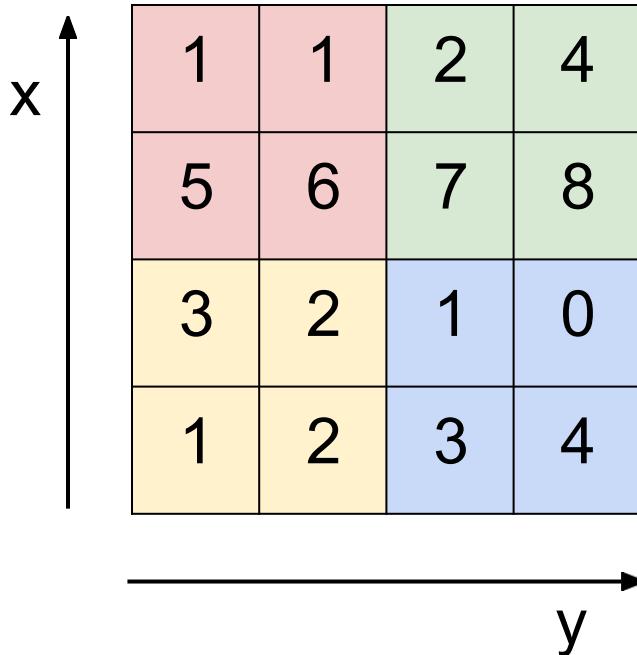
car  
truck  
airplane  
ship  
horse

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

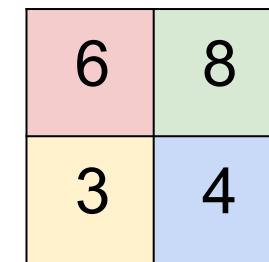


# MAX POOLING

Single depth slice



max pool with 2x2 filters  
and stride 2



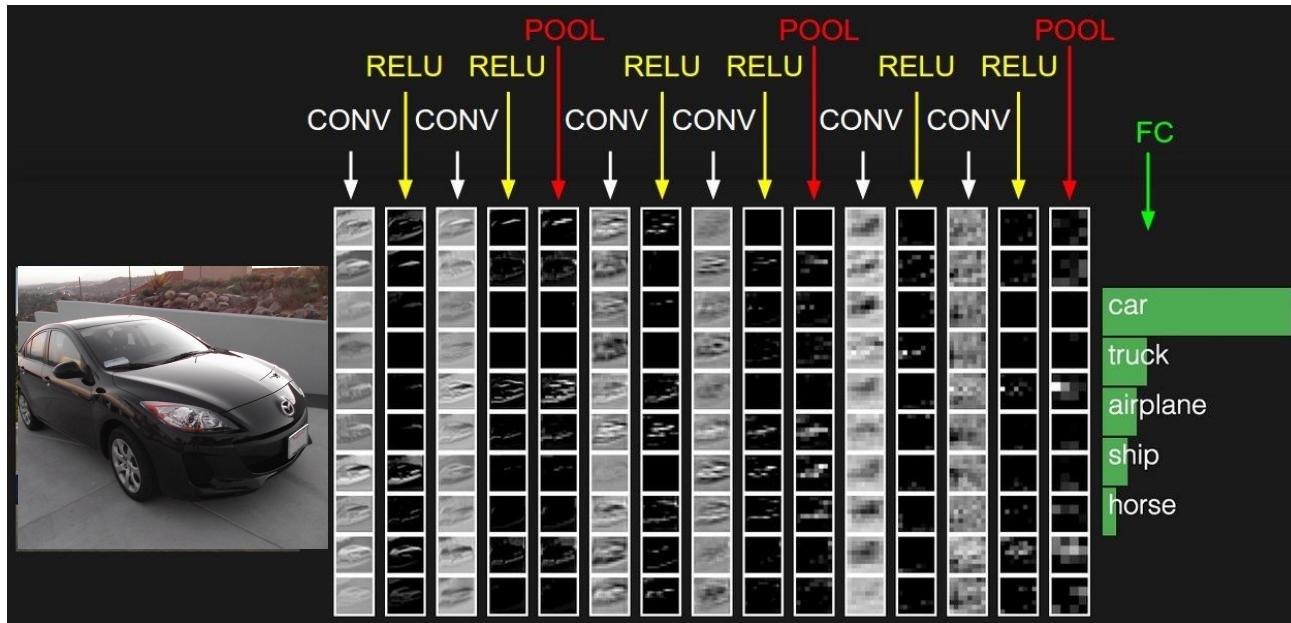
- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

## Common settings:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$   $F = 2, S = 2$
- Requires three hyperparameters:
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

# Fully Connected Layer (FC layer)

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



# Summary

- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like  
 **$[(\text{CONV-RELU})^* \text{N-POOL?}]^* \text{M-(FC-RELU)}^* \text{K,SOFTMAX}$**   
where N is usually up to ~5, M is large,  $0 \leq K \leq 2$ .
  - but recent advances such as ResNet/GoogLeNet challenge this paradigm

# CNN Architectures

## Case Studies

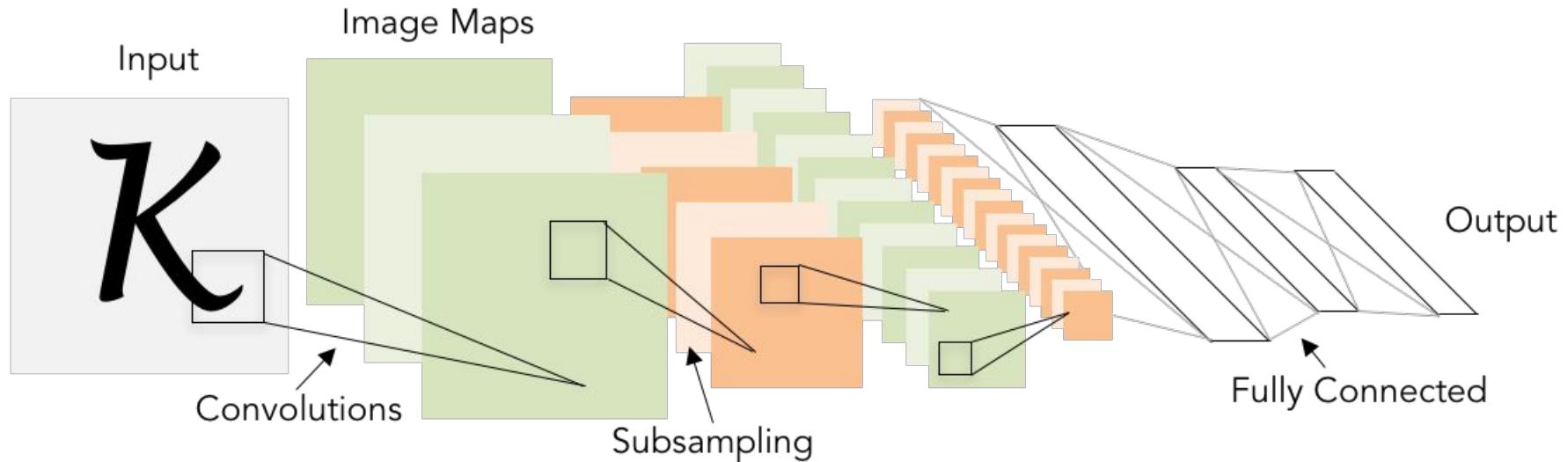
- AlexNet
- VGG
- GoogLeNet
- ResNet

## Also....

- NiN (Network in Network)
- Wide ResNet
- ResNeXT
- Stochastic Depth
- DenseNet
- FractalNet
- SqueezeNet

# Review: LeNet-5

[LeCun et al., 1998]

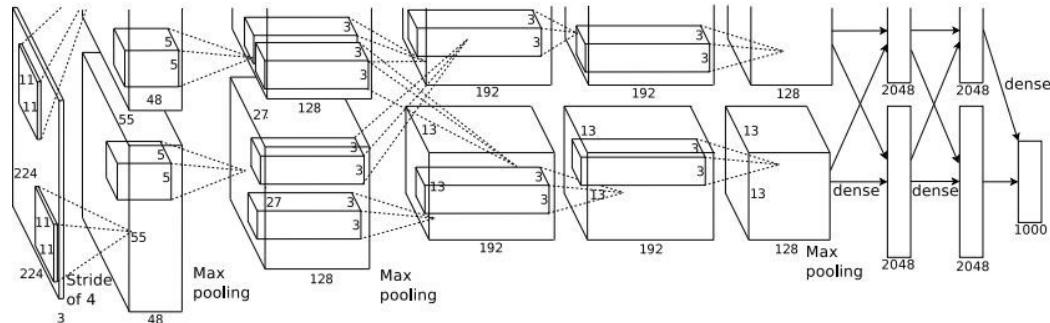


Conv filters were 5x5, applied at stride 1

Subsampling (Pooling) layers were 2x2 applied at stride 2  
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

# Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

**First layer (CONV1):** 96 11x11 filters applied at stride 4

=>

**Output volume [55x55x96]**

Parameters:  $(11 \times 11 \times 3) \times 96 + 96 = 35K$

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

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

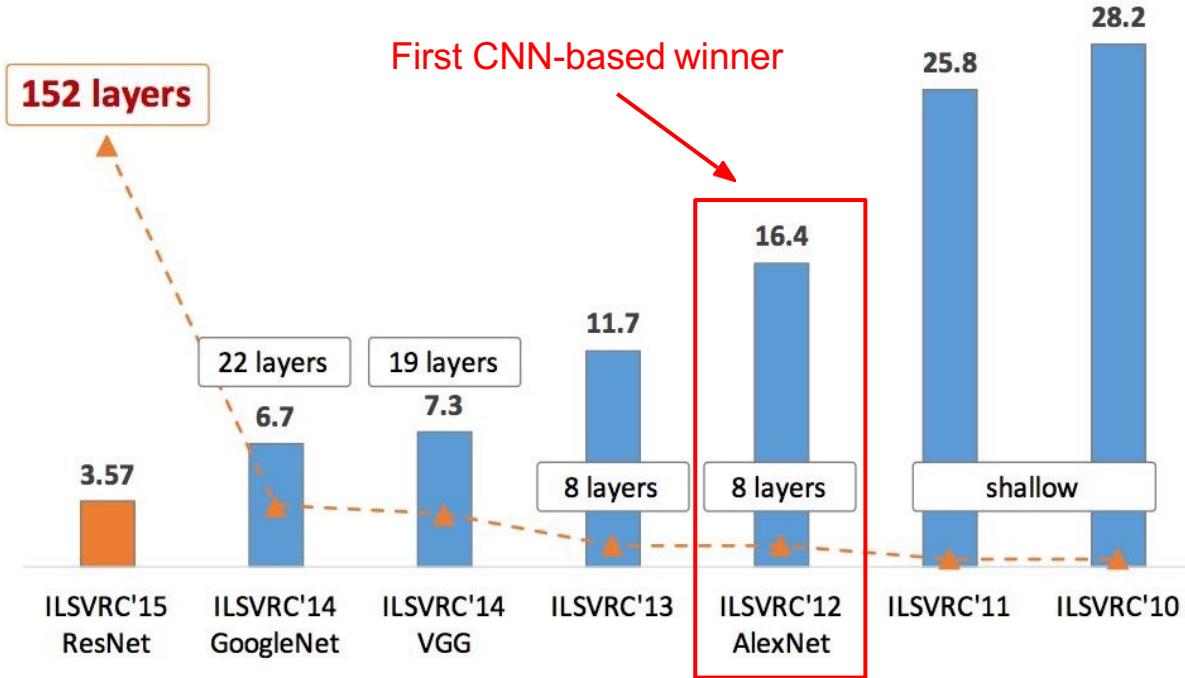


Figure copyright Kaiming He, 2016. Reproduced with permission.

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

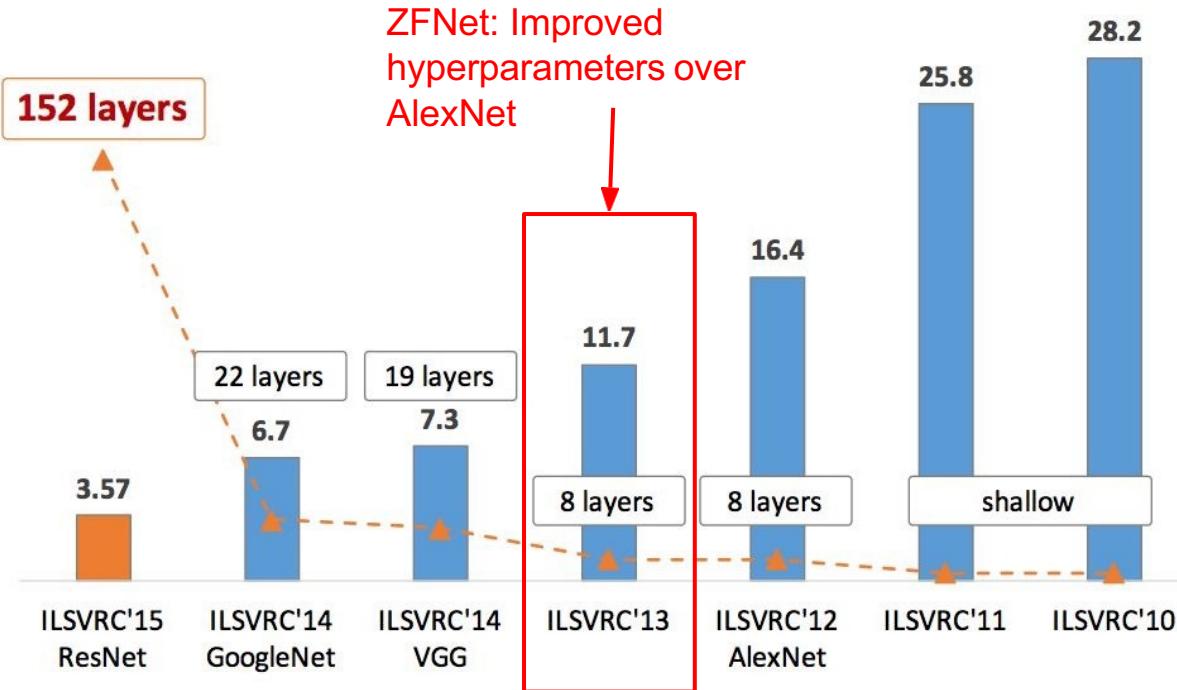
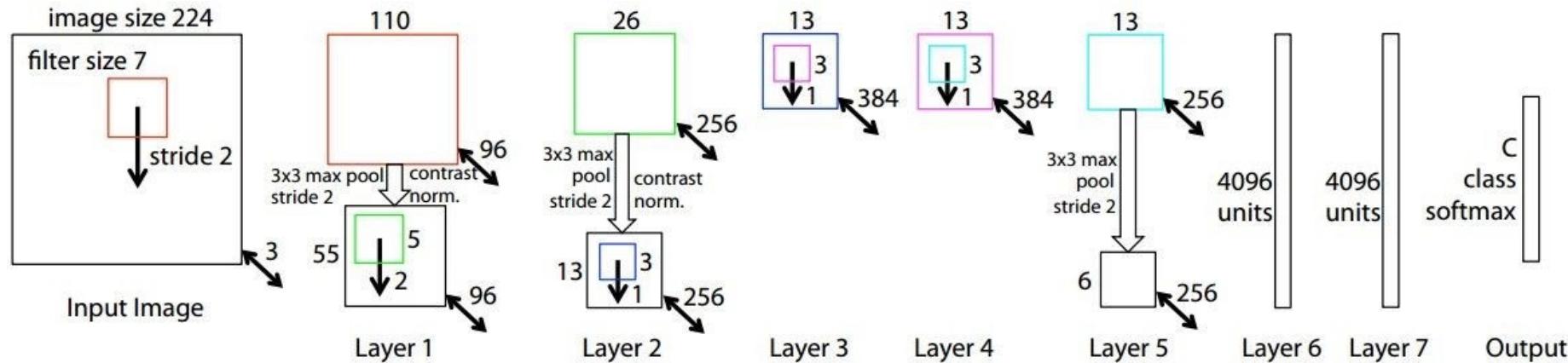


Figure copyright Kaiming He, 2016. Reproduced with permission.

# ZFNet

[Zeiler and Fergus, 2013]



AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

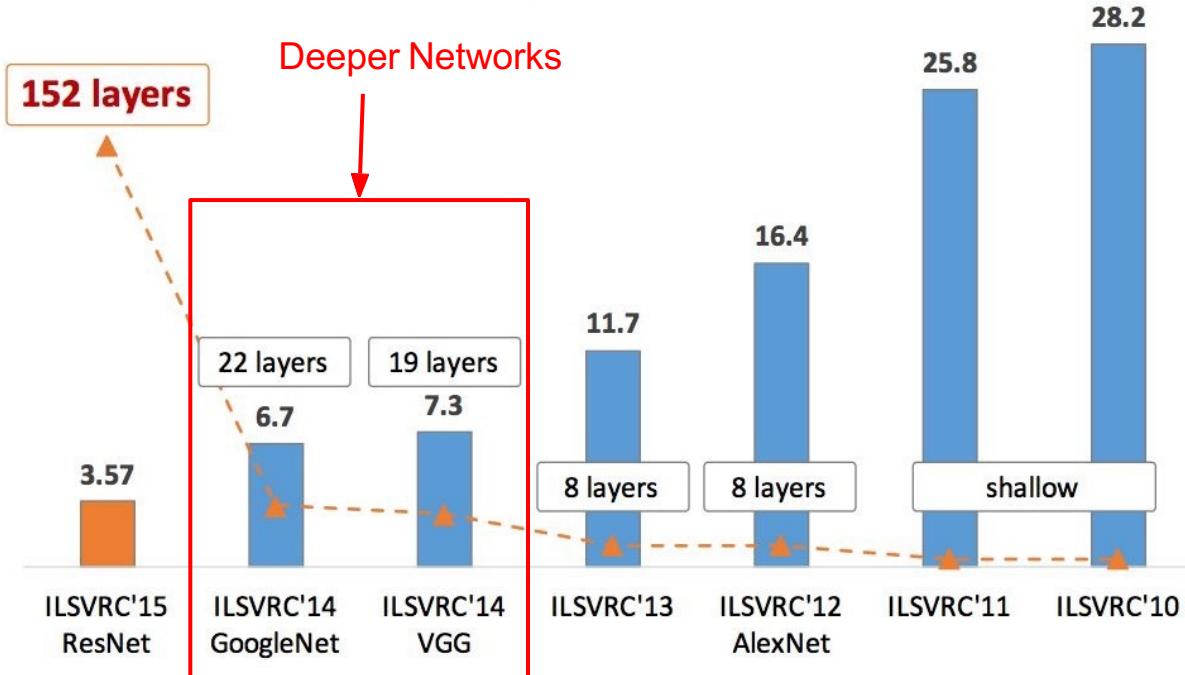


Figure copyright Kaiming He, 2016. Reproduced with permission.

# Case Study: VGGNet

*[Simonyan and Zisserman, 2014]*

Small filters, Deeper networks

8 layers (AlexNet)

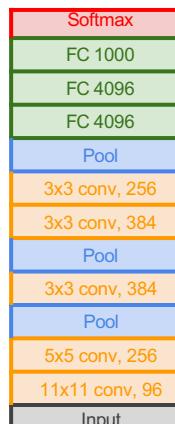
-> 16 - 19 layers (VGG16Net)

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

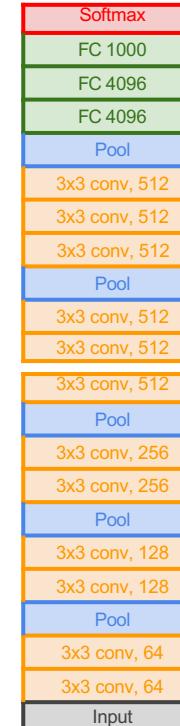
11.7% top 5 error in ILSVRC'13

(ZFNet)

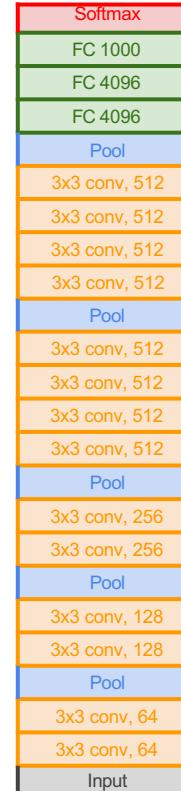
-> 7.3% top 5 error in ILSVRC'14



AlexNet



VGG16



VGG19

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

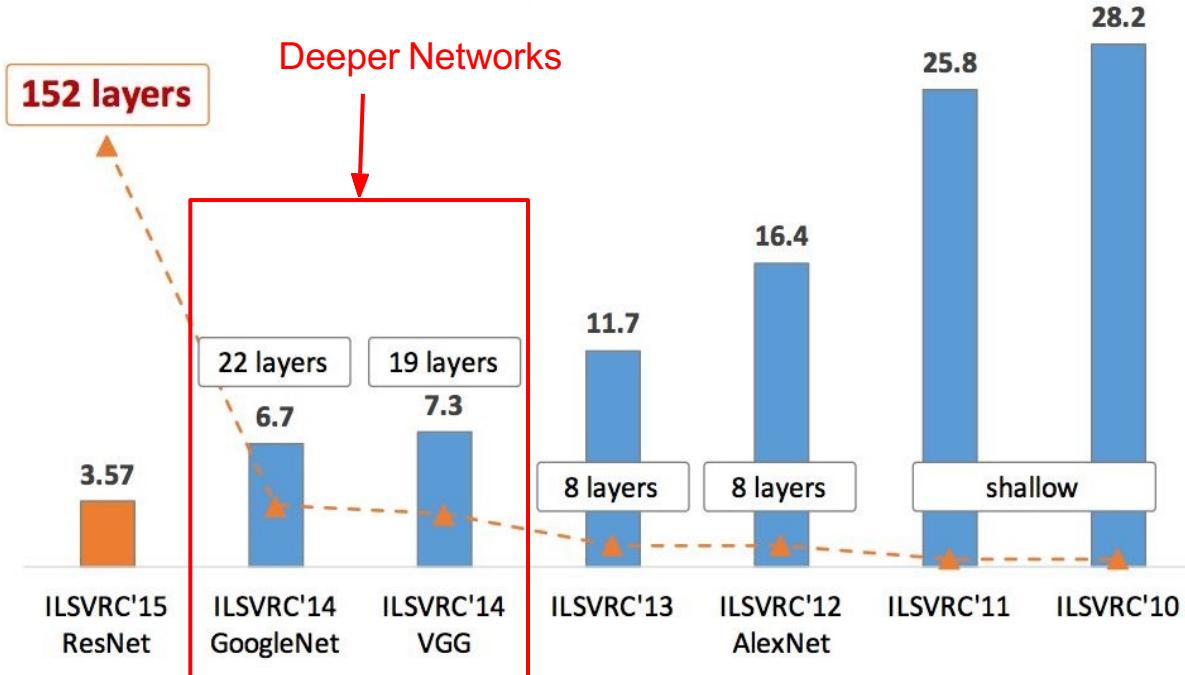


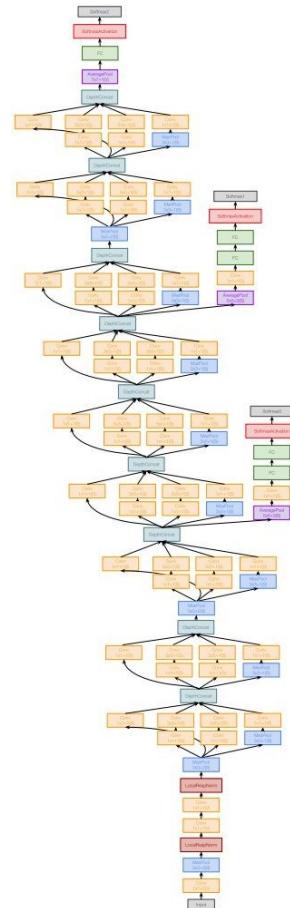
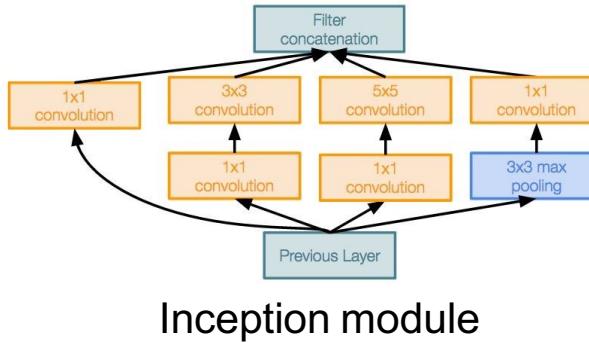
Figure copyright Kaiming He, 2016. Reproduced with permission.

# Case Study: GoogLeNet

[Szegedy et al., 2014]

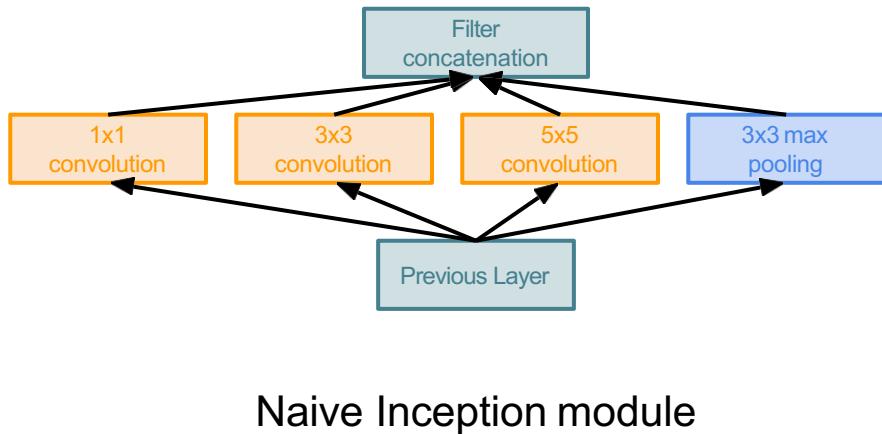
Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!  
12x less than AlexNet
- ILSVRC’14 classification winner  
(6.7% top 5 error)



# Case Study: GoogLeNet

[Szegedy et al., 2014]



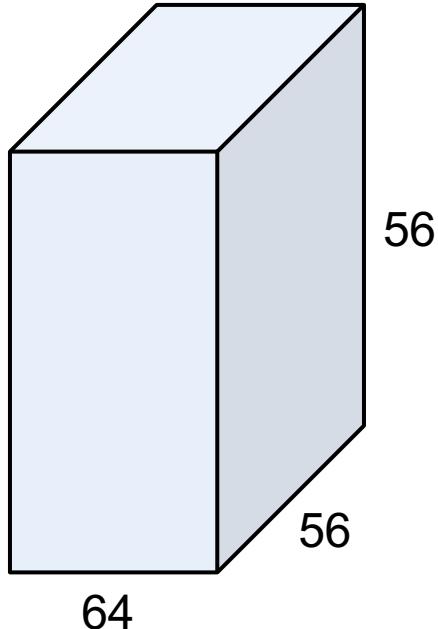
Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution ( $1 \times 1$ ,  $3 \times 3$ ,  $5 \times 5$ )
- Pooling operation ( $3 \times 3$ )

Concatenate all filter outputs together depth-wise

Q: What is the problem with this?  
[Hint: Computational complexity]

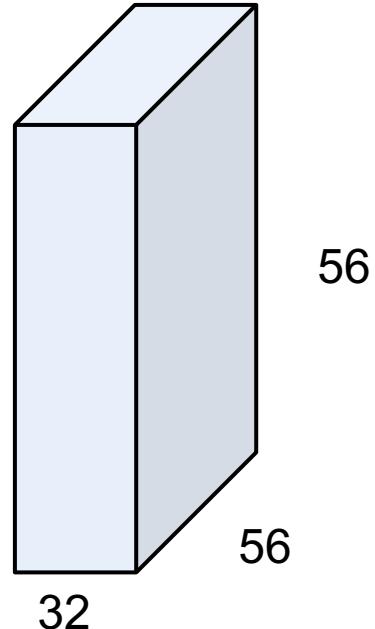
# Reminder: 1x1 convolutions



1x1 CONV  
with 32 filters

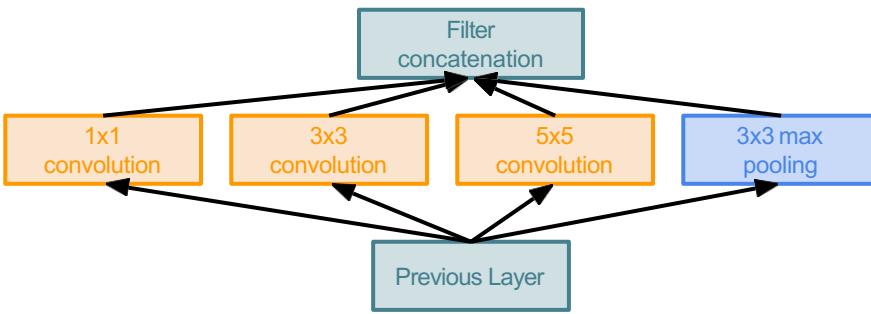
---

(each filter has size  
1x1x64, and performs a  
64-dimensional dot  
product)

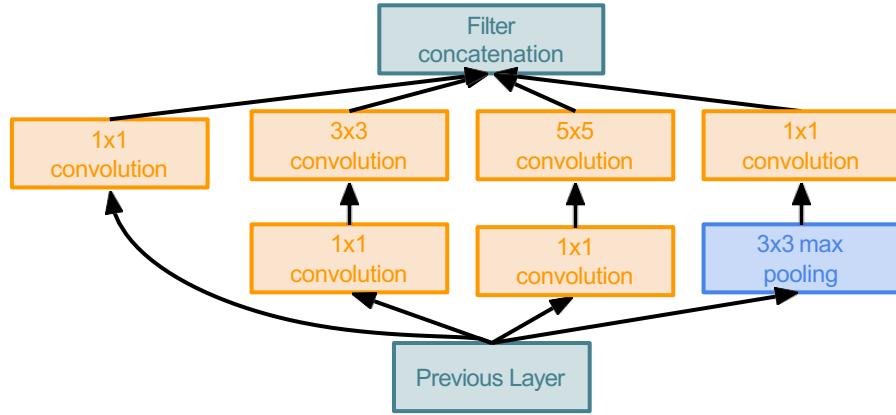


# Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module



Inception module with dimension reduction

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

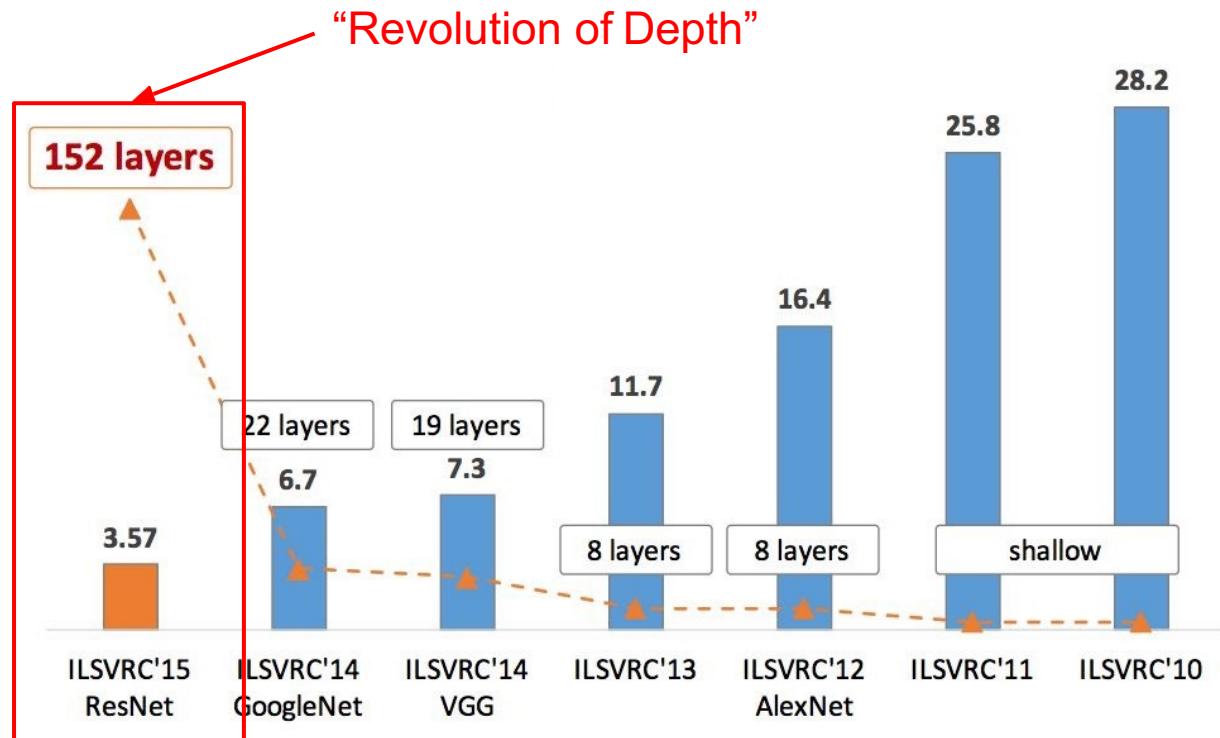


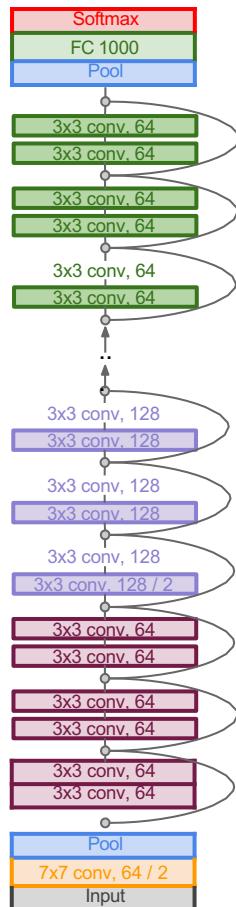
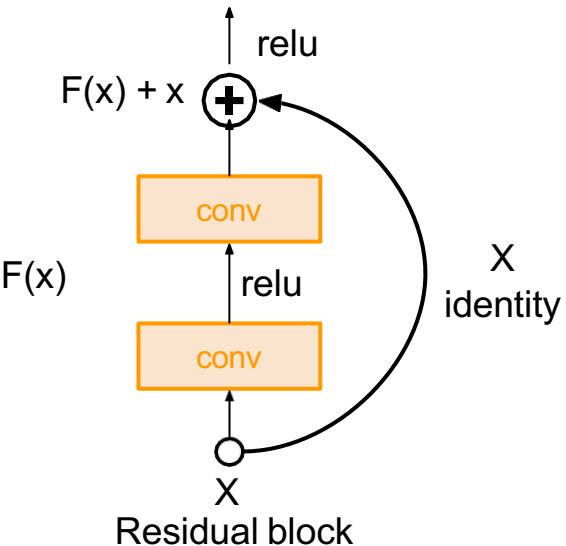
Figure copyright Kaiming He, 2016. Reproduced with permission.

# Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

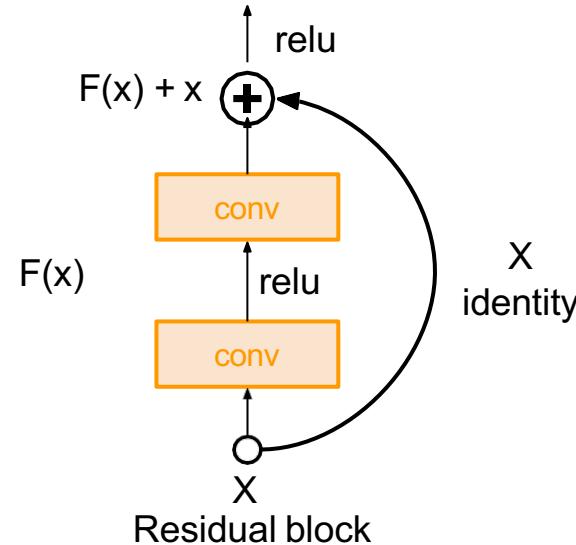
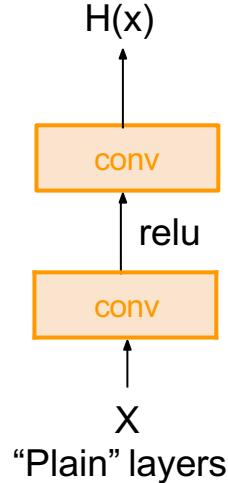
- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



# Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



# Case Study: ResNet

## [He et al., 2015]

### Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lower training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

### MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places** in all five main tracks
  - ImageNet Classification: “Ultra-deep” (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

# Case Study: ResNet

## [He et al., 2015]

### Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowing training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

### MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places** in all five main tracks
  - ImageNet Classification: “Ultra-deep” (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than “human performance”! (Russakovsky 2014)

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

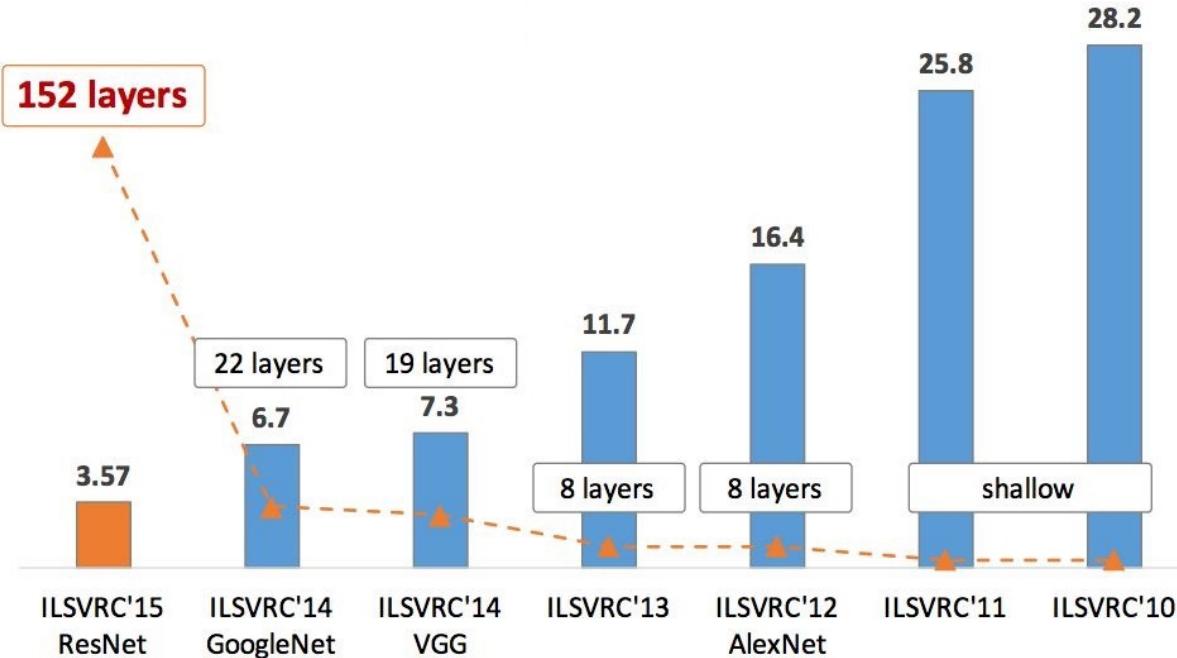
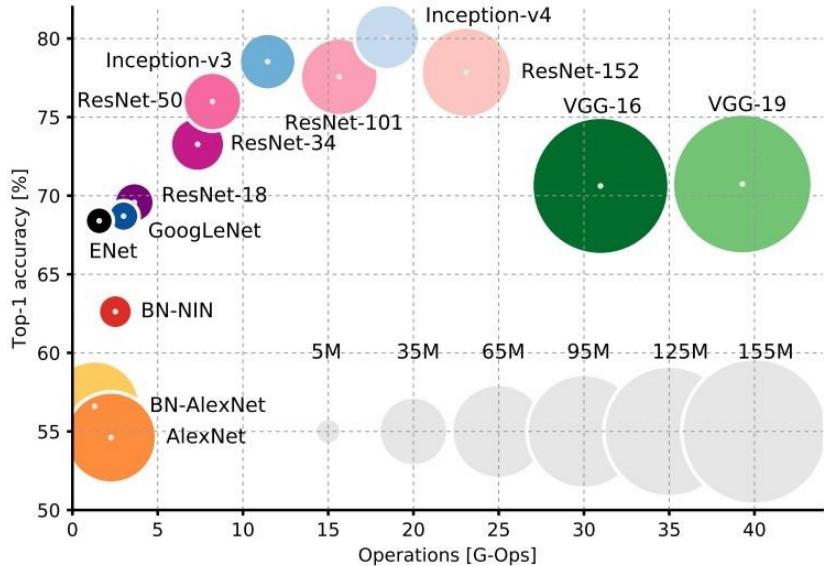
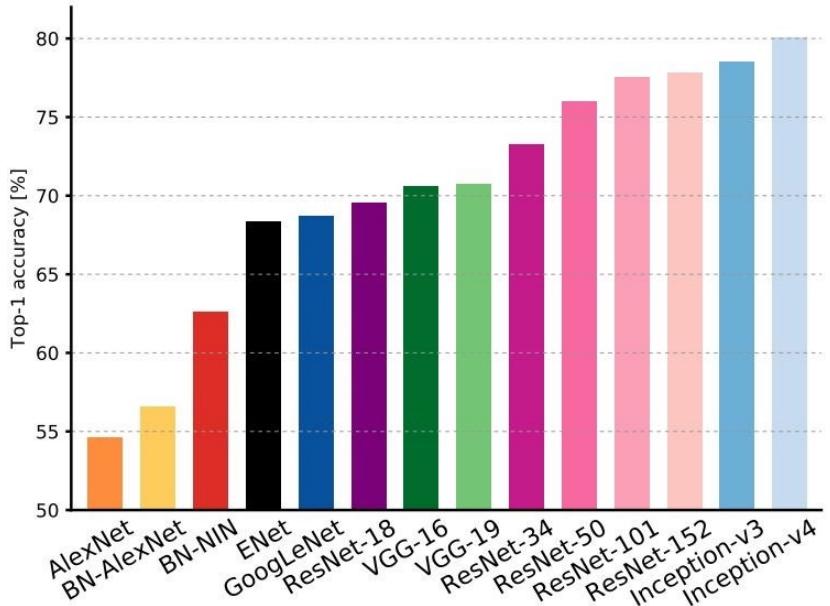


Figure copyright Kaiming He, 2016. Reproduced with permission.

# Comparing complexity...



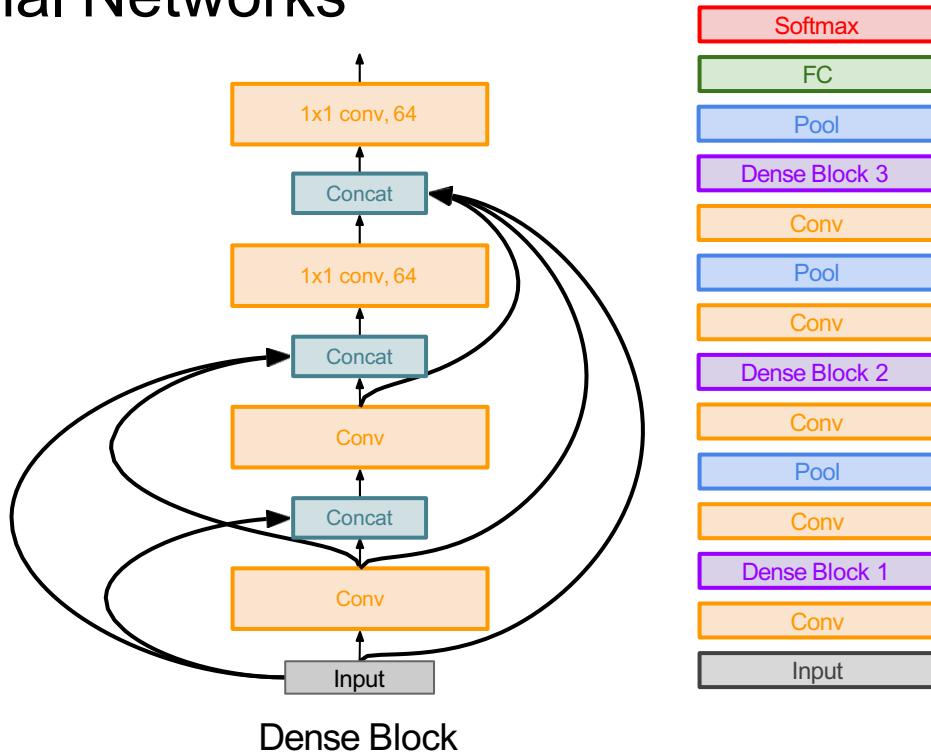
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

# Beyond ResNets...

## Densely Connected Convolutional Networks

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



Efficient networks...

# SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size

[Iandola et al. 2017]

- Fire modules consisting of a ‘squeeze’ layer with 1x1 filters feeding an ‘expand’ layer with 1x1 and 3x3 filters
- AlexNet level accuracy on ImageNet with 50x fewer parameters
- Can compress to 510x smaller than AlexNet (0.5Mb)

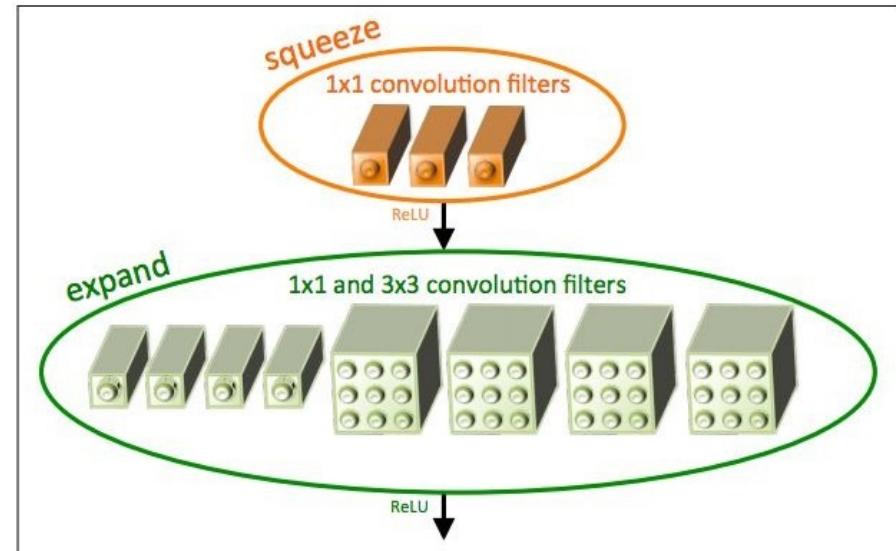
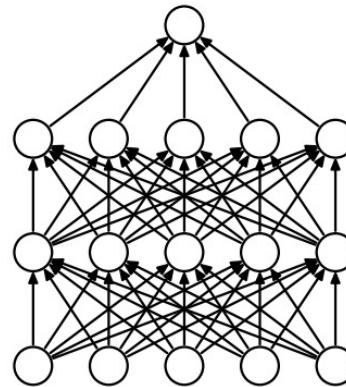


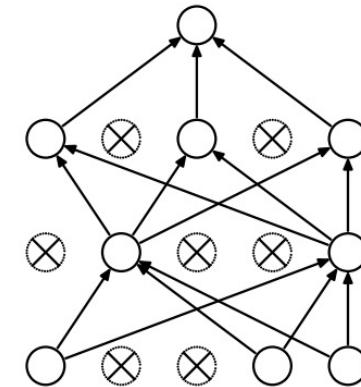
Figure copyright Iandola, Han, Moskewicz, Ashraf, Dally, Keutzer, 2017. Reproduced with permission.

# Regularizations:

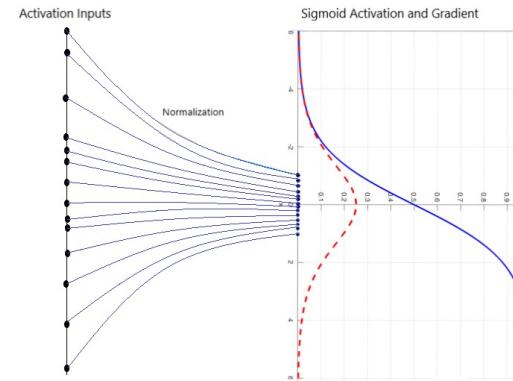
- Dropout
  - Randomly drop units (50%)
  - prevents overfitting
- Batch Normalization
  - Normalize each layer inputs
  - Sometimes can replace Dropout



(a) Standard Neural Net



(b) After applying dropout.



# Summary: CNN Architectures

## Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

## Also....

- NiN (Network in Network)
- Wide ResNet
- ResNeXT
- Stochastic Depth
- DenseNet
- FractalNet
- SqueezeNet

# Summary: CNN Architectures

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- ResNet current best default
- Trend towards extremely deep networks
- Significant research centers around design of layer / skip connections and improving gradient flow
- Even more recent trend towards examining necessity of depth vs. width and residual connections