Lecture

Convolutional Neural Networks

CNNs

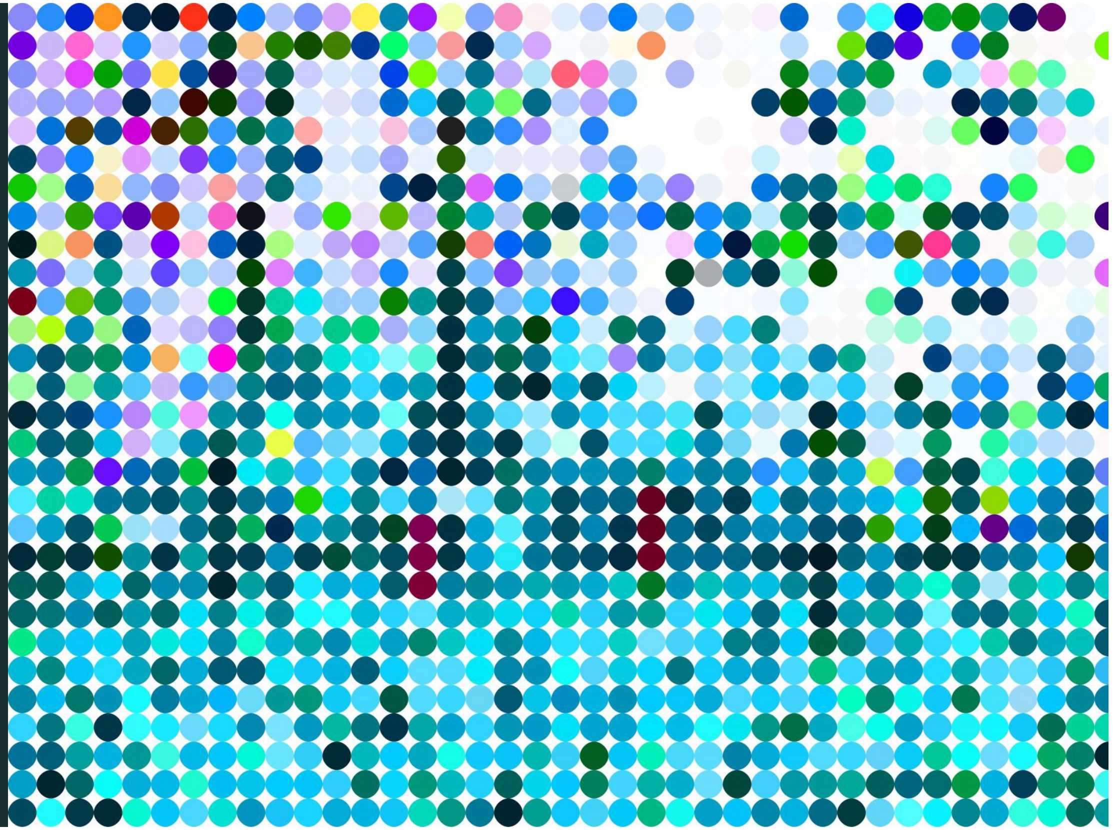
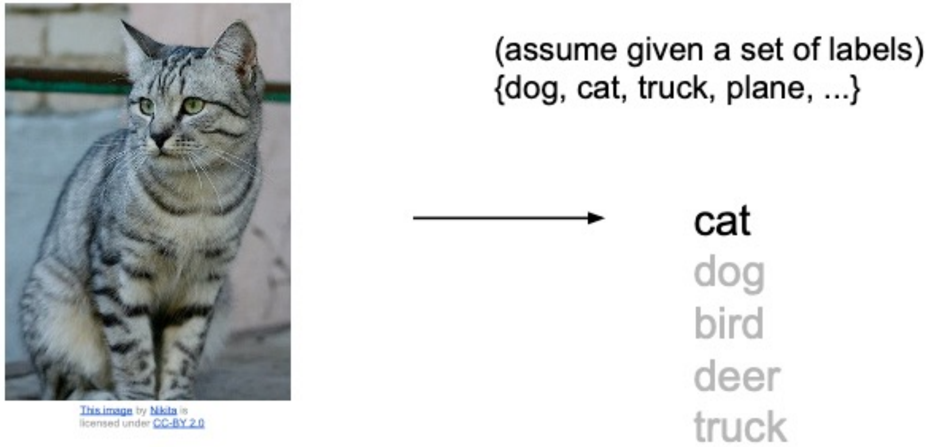


Image Classification

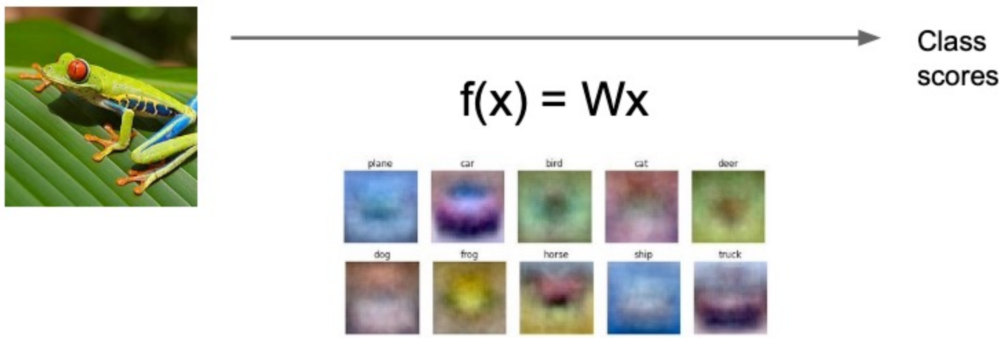
- A core task in Computer Vision
- Trained kernels are powering the network
- Add a regression branch to the classifcation network and you have a object detection network.



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Pixel Space

- "Global" class-wise kernels is not that versatile
  - Weak generalization
  - No domain



Global Kernels

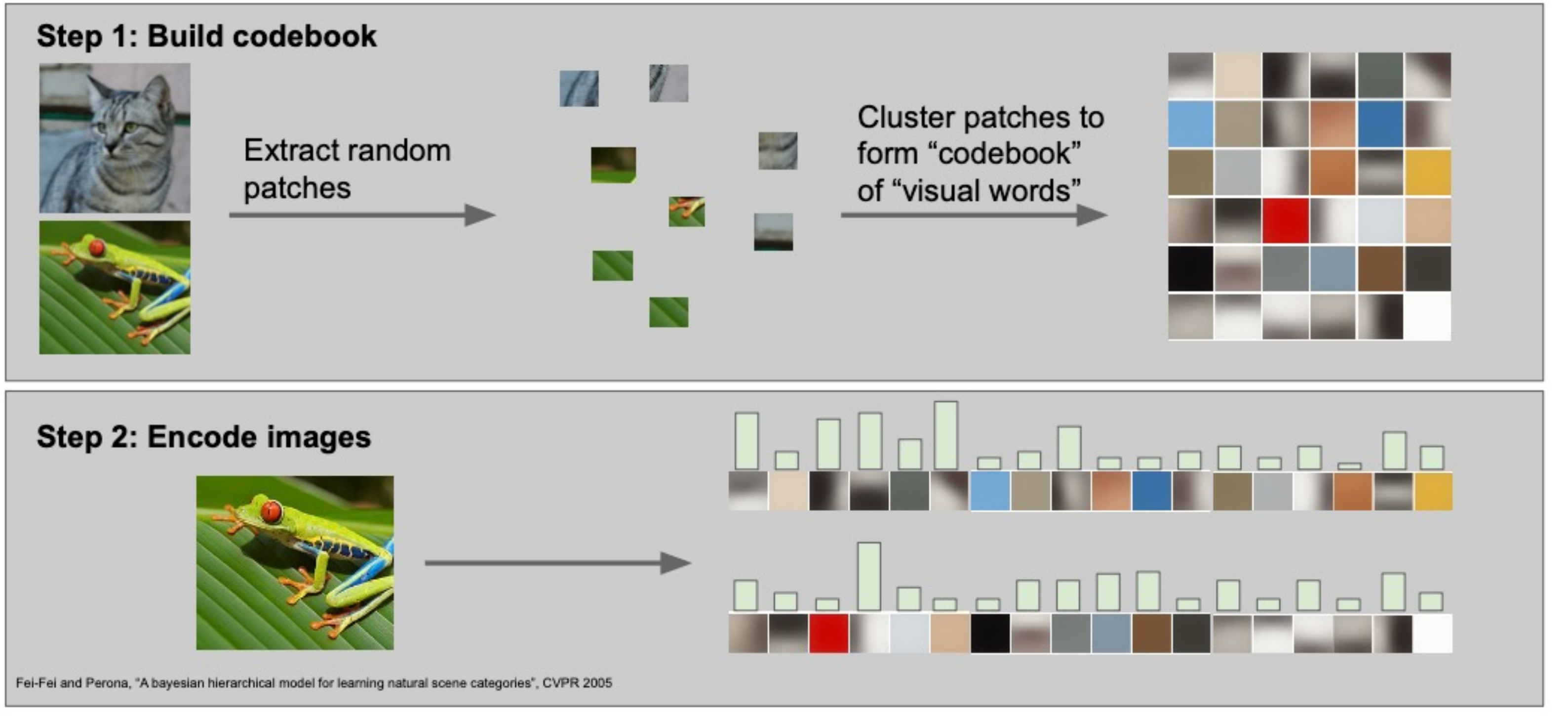
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Image Features

- Another approach:
  - Image Features
    - Classification
  - Object Features
    - Object Detection

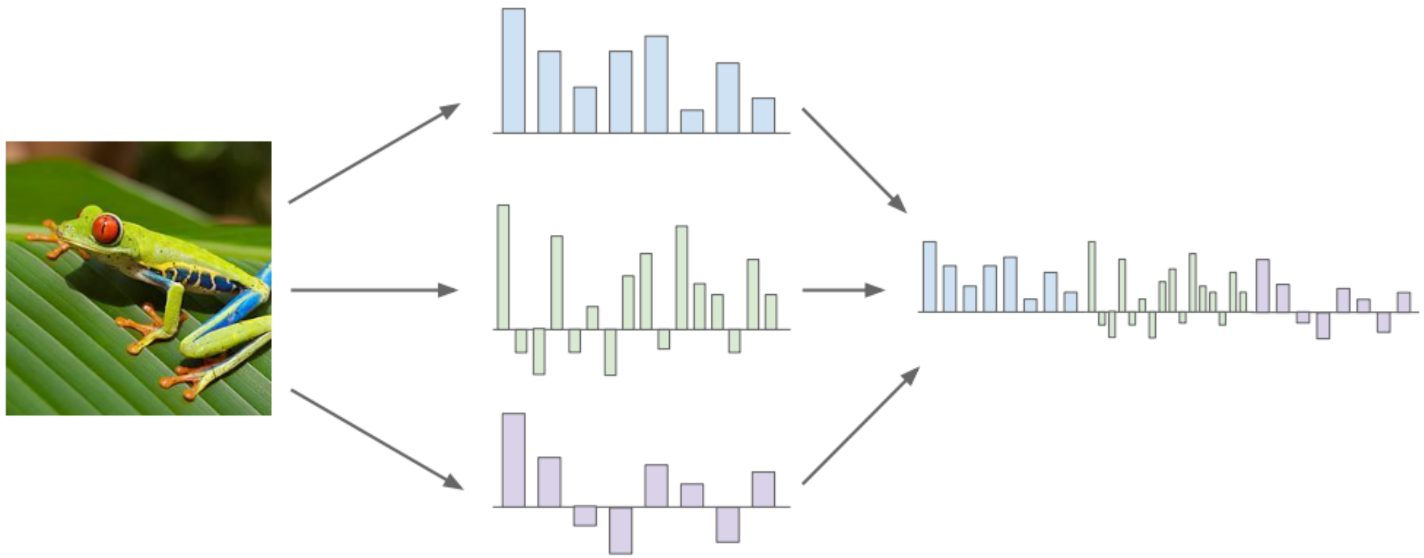
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Example: Codebook of Visual Patterns



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

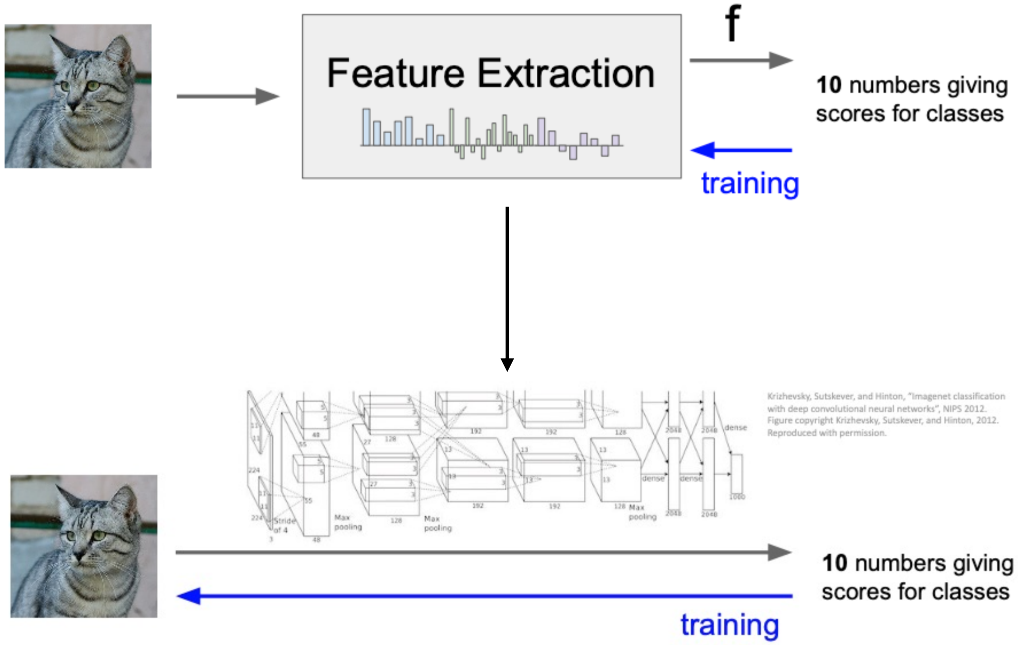
What if we can learn the features?



Then we have a unique fingerprint for each class.
Ideally easy to distinguish for an MLP

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Image Features vs. ConvNets



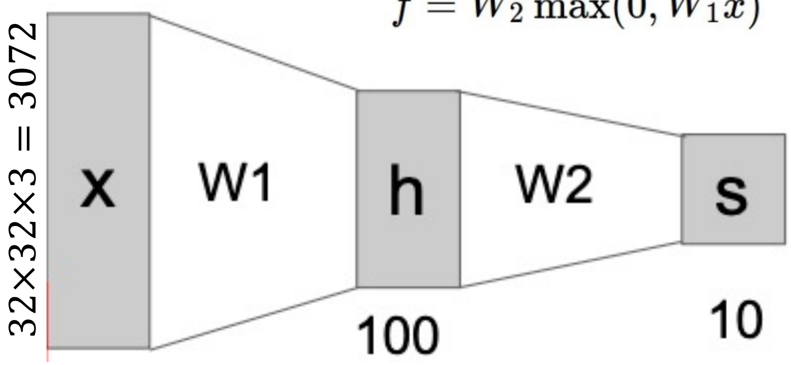
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

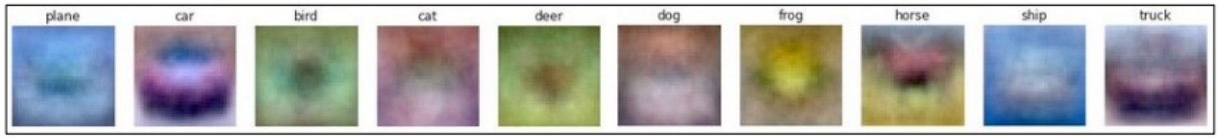
Neural Networks, why not?

- Computationally Reassource Heavy
- Spatial Structure of the image is destroyed
    - Locational Invariance is important
        - Object Detection
        - Pattern in feature location (classification)
            - Washing this out is removing information

$$f = Wx$$

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





Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolutional Neural Networks

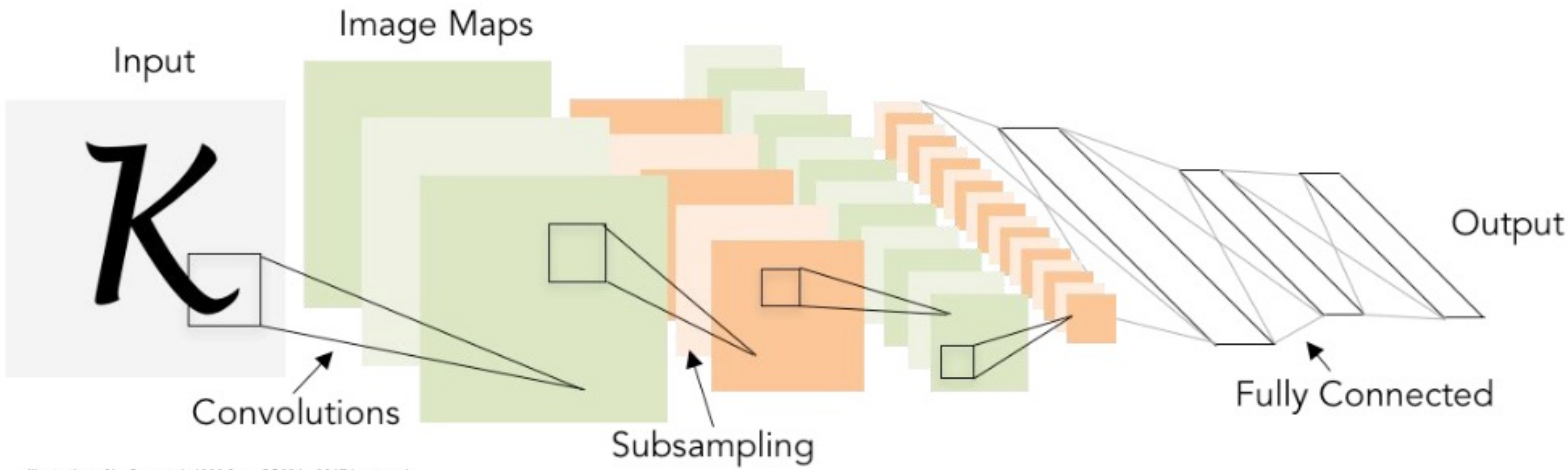
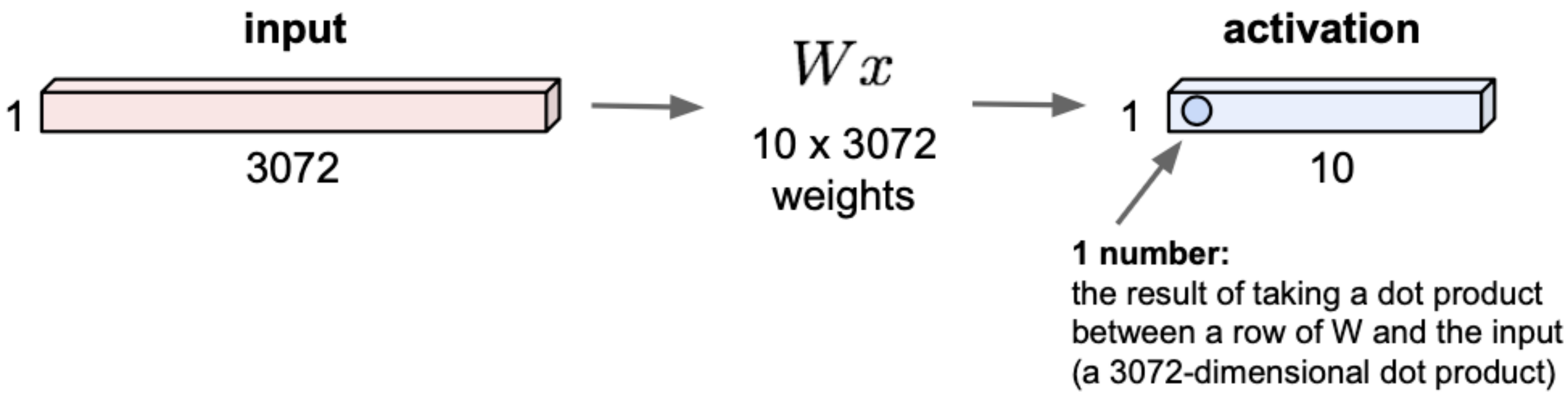


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

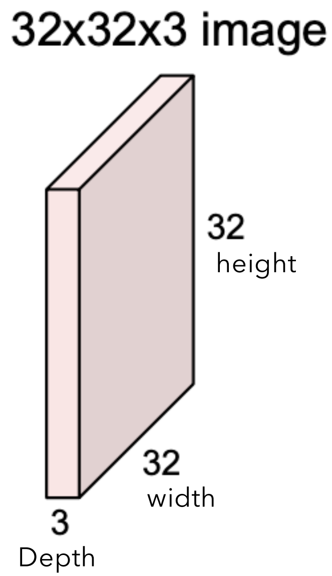
Fully Connected Layer (recap)

$32 \times 32 \times 3$ $Image\ stretched\ (flattened) to\ 3072 \times 1$



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

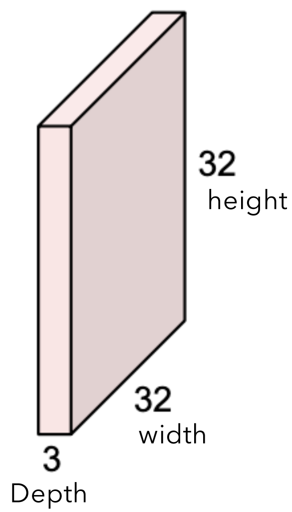
Convolution Layer



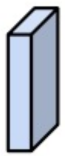
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

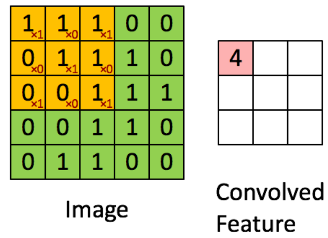
Convolution Layer

32x32x3 image



5x5x3 filter



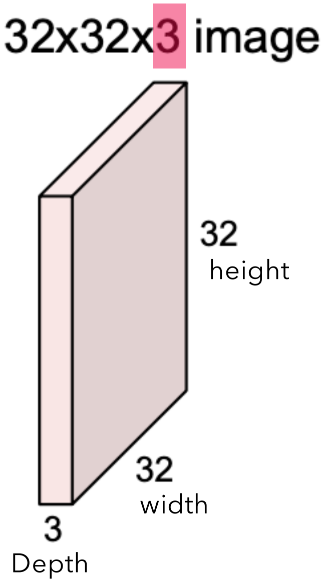


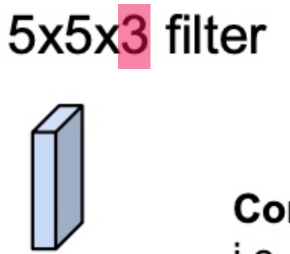
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Remember: we want to preserve spatial structure

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer





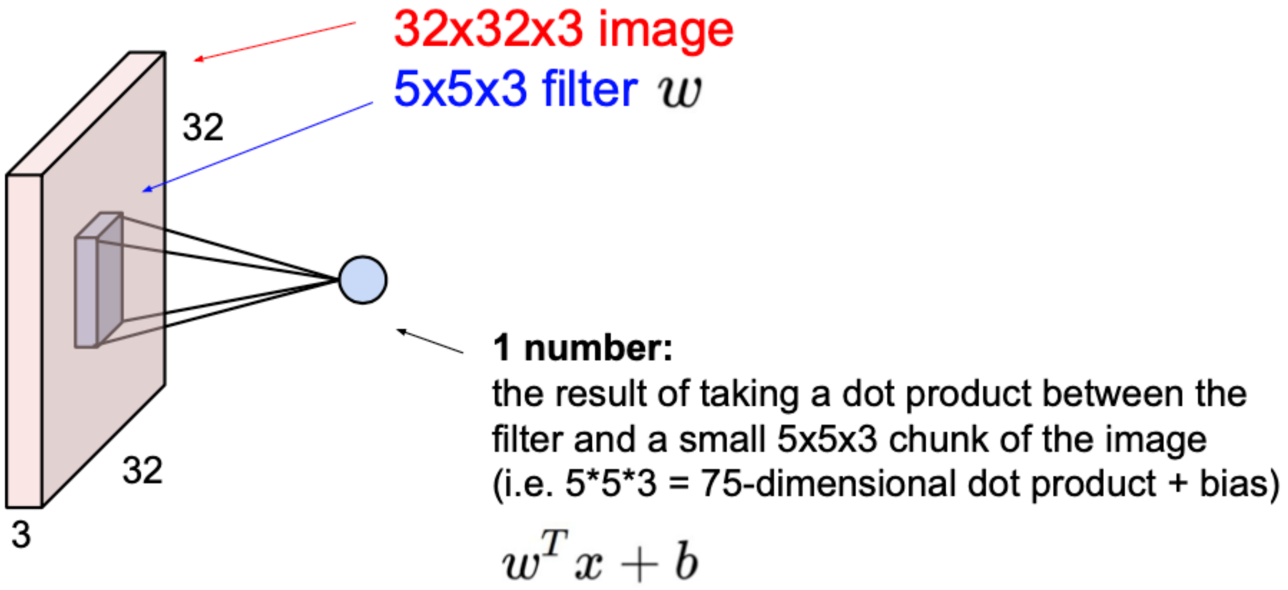
Filter always extend the full depth of the input volume

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

Remember: we want to preserve spatial structure

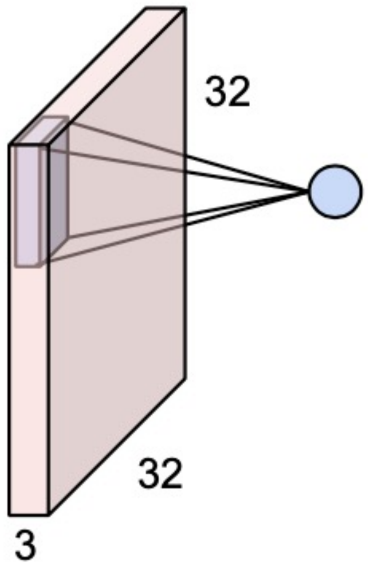
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer



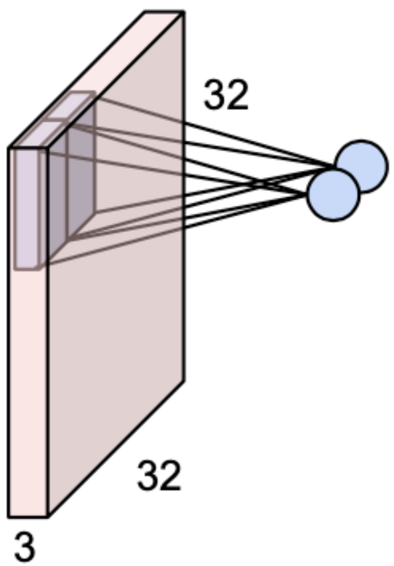
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer



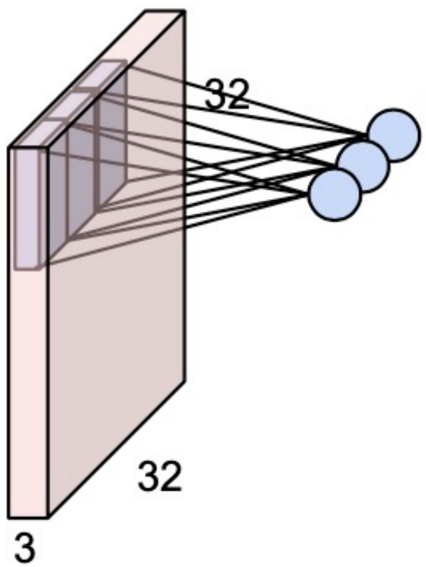
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer



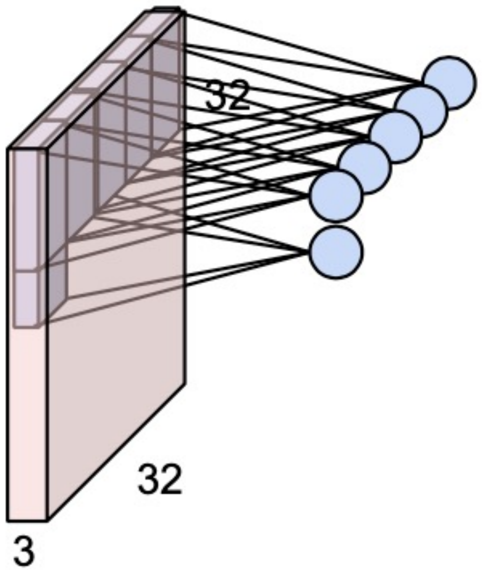
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer



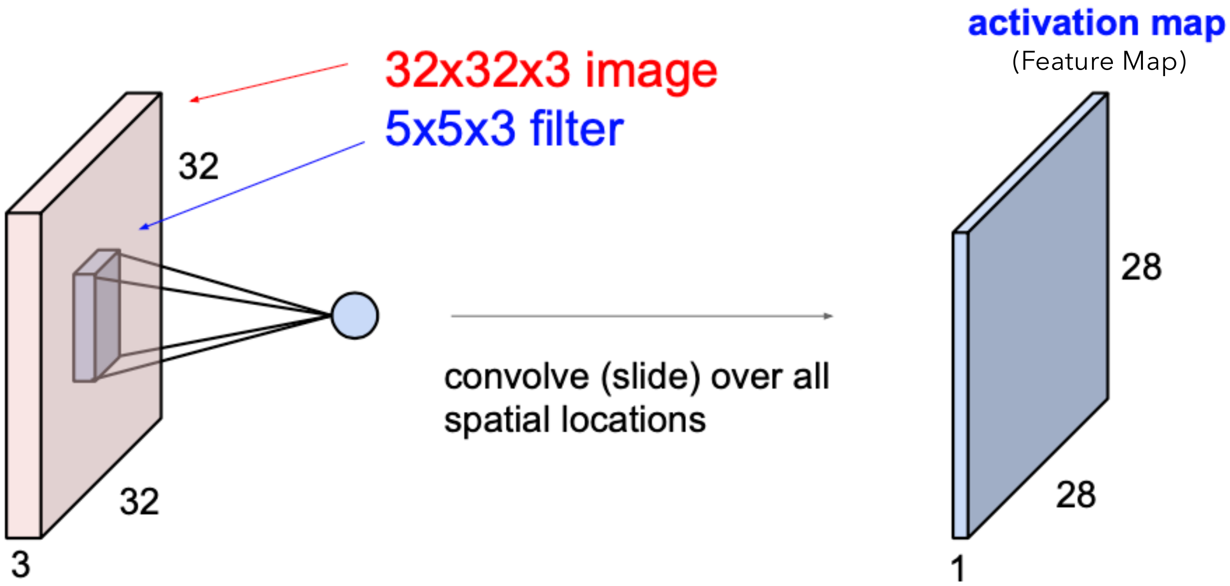
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer



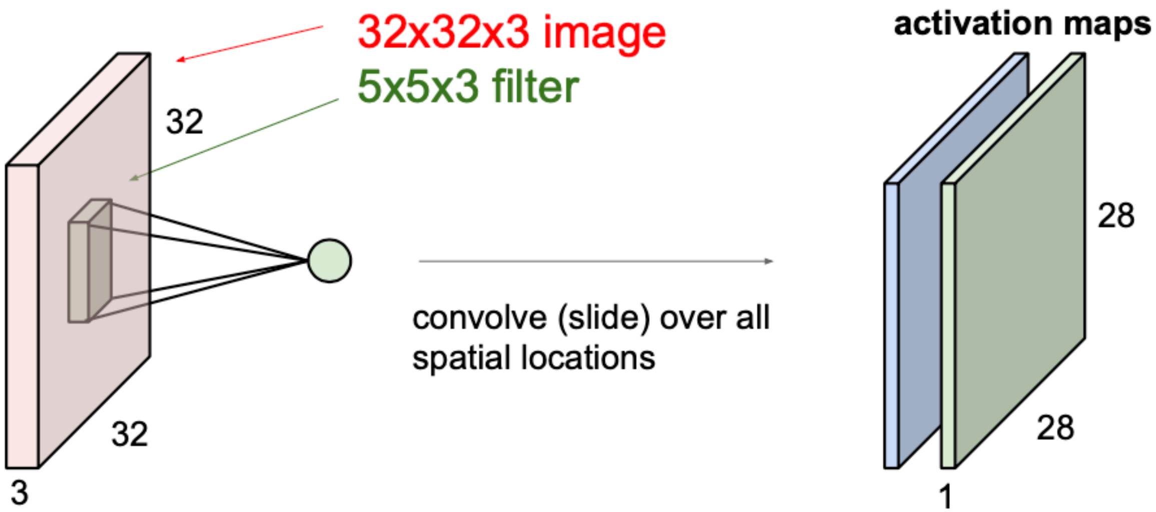
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

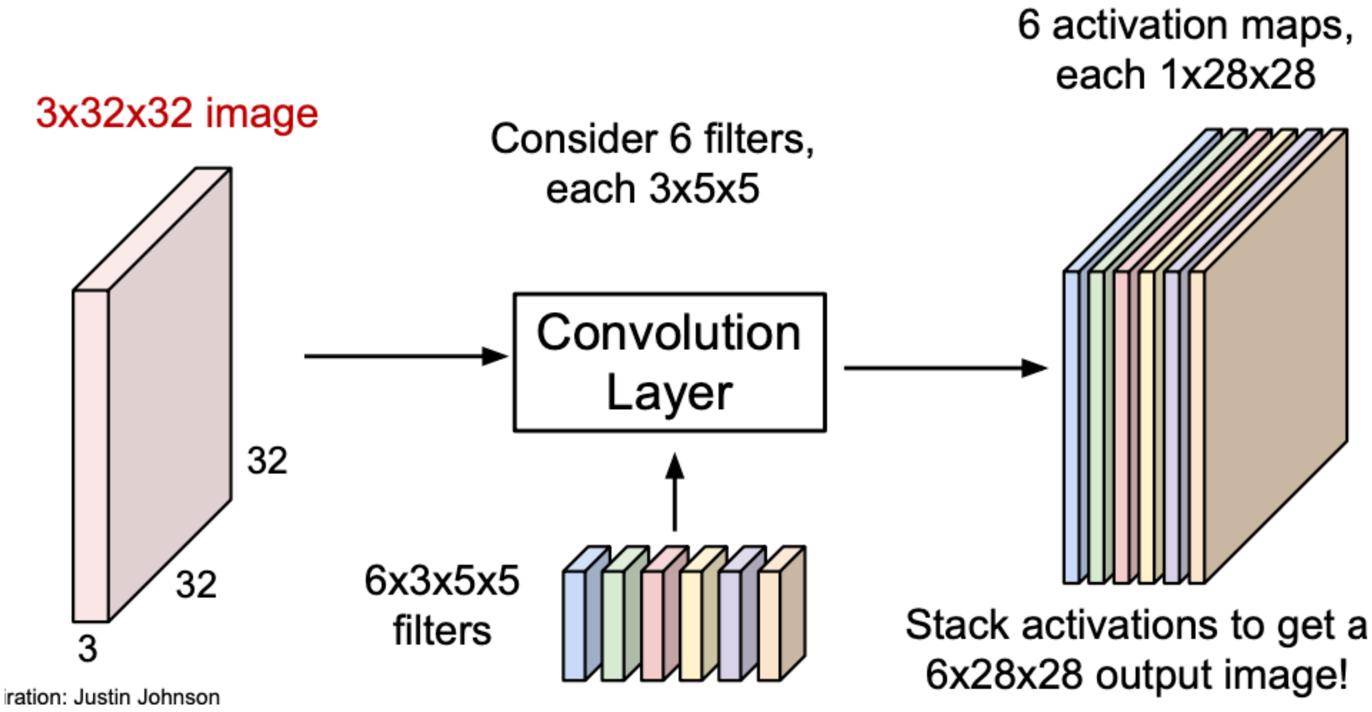
Convolution Layer



Consider the green a second filter

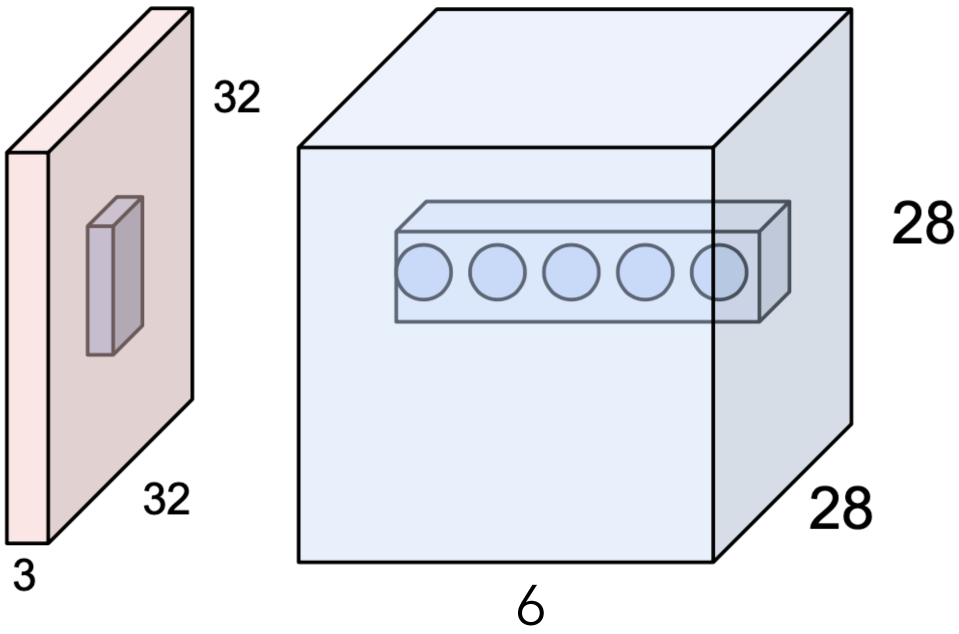
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

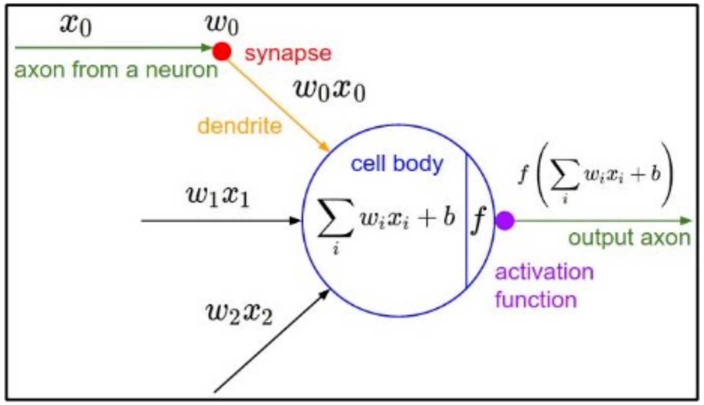
Convolution Layer



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer – *neurons*





- CONV Layers consists of neurons arranged into a 3D Grid (28x28x6)
- There will be 6 different neurons looking at the same region in the input volume

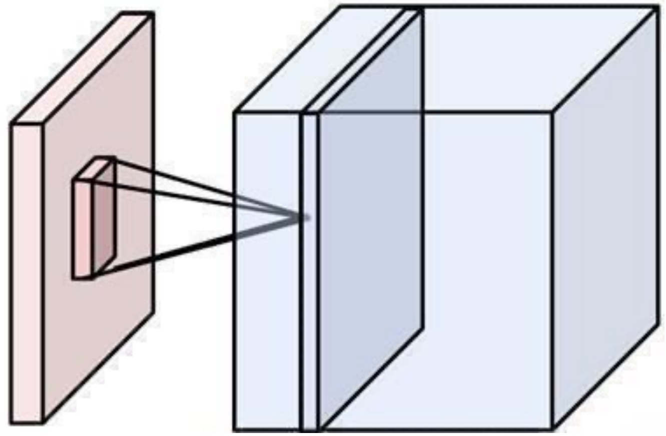
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer

- **Local** Connectivity
  - Neurons in a layer are only connected to a small region of the layer before it

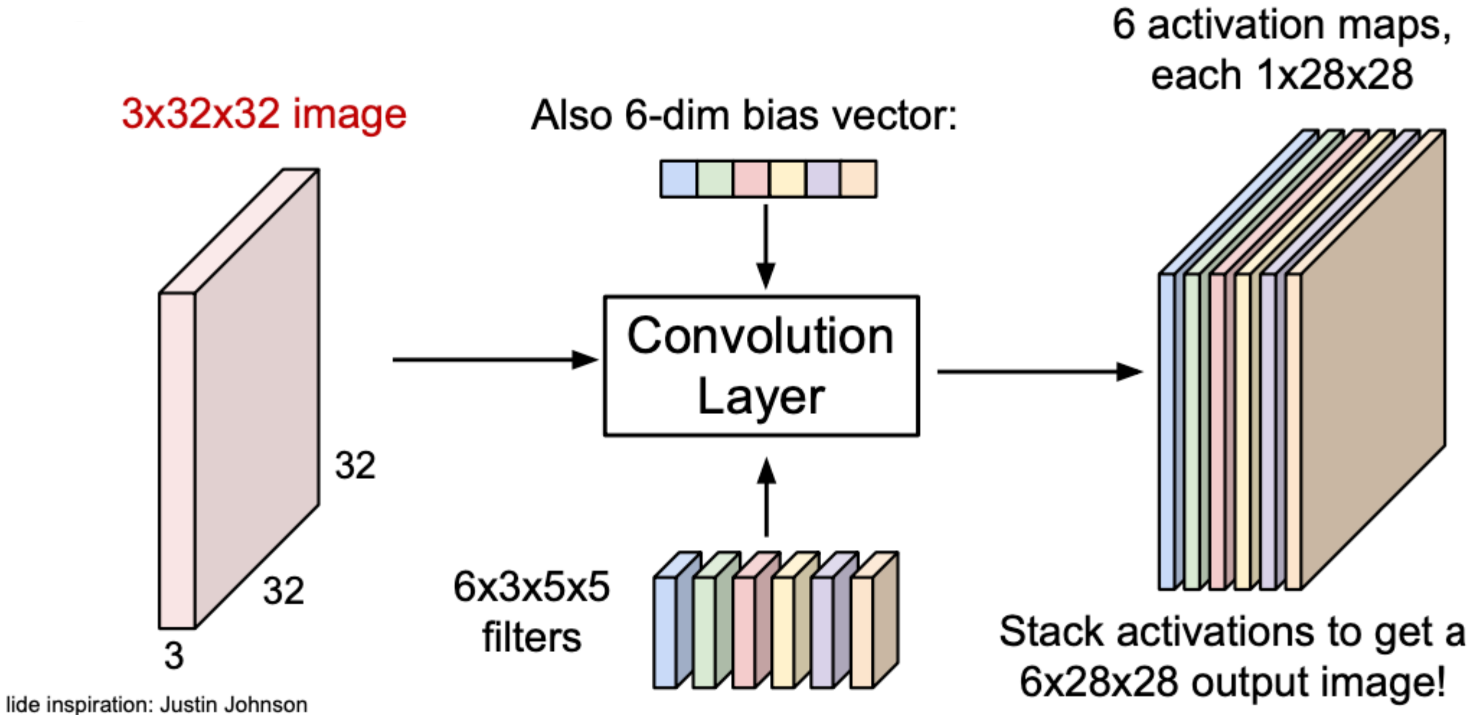
- **Share** weight parameters accros spatial positions
  - The same kernel (with fixed weights) is convoled over the whole input.
  - Factor in translational variance

    "If you have a car in the image, whether this is in top-left or bottom-right, the filters (which learn to extract certain features) or produce a higher activation once they see a similar pattern that they learned regardless of the location of the feature"



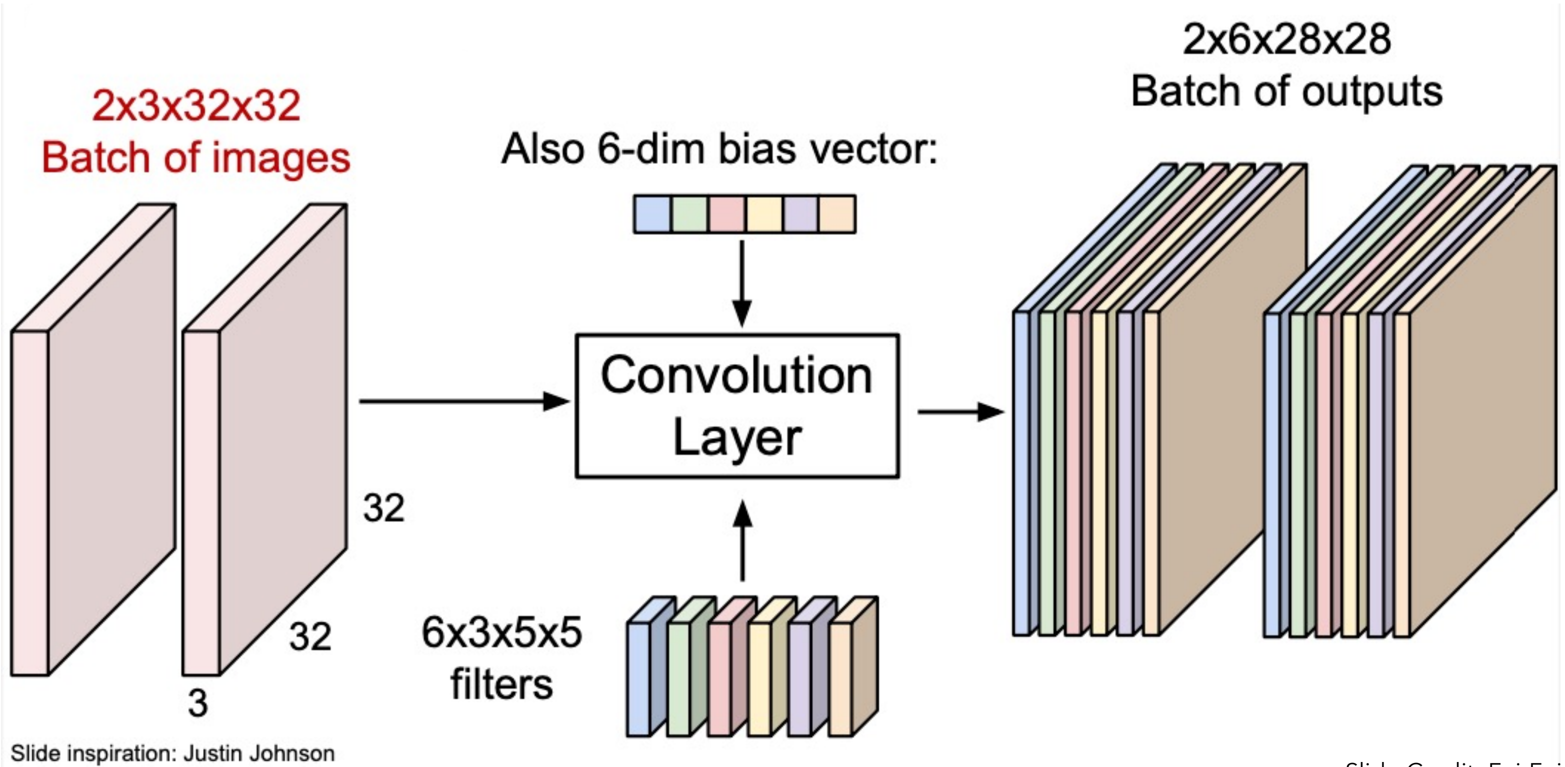
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer



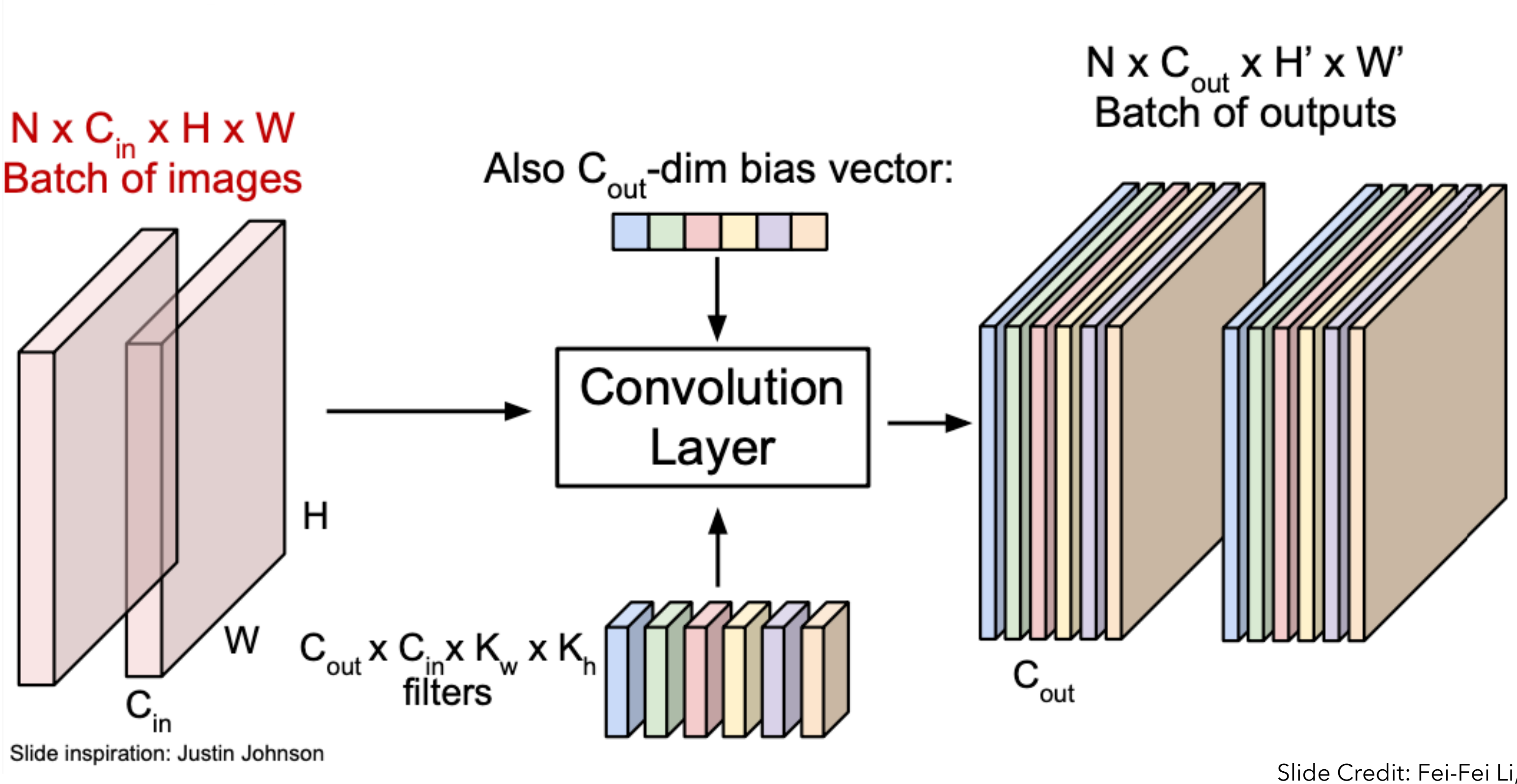
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer - Batches



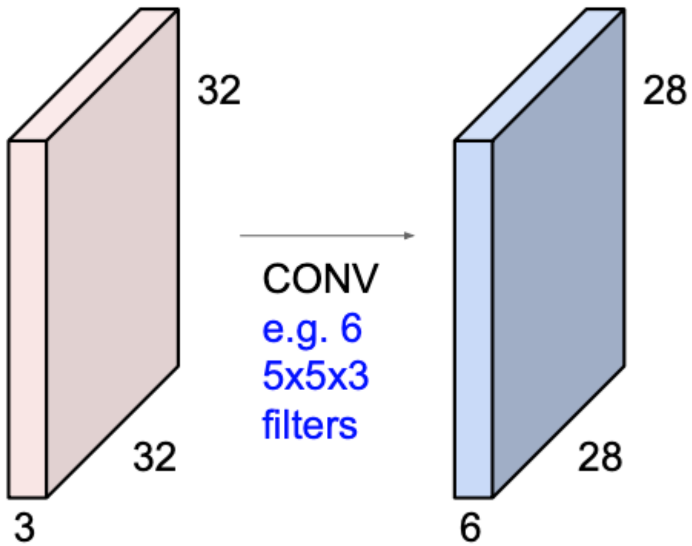
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolution Layer – Batches Generalized



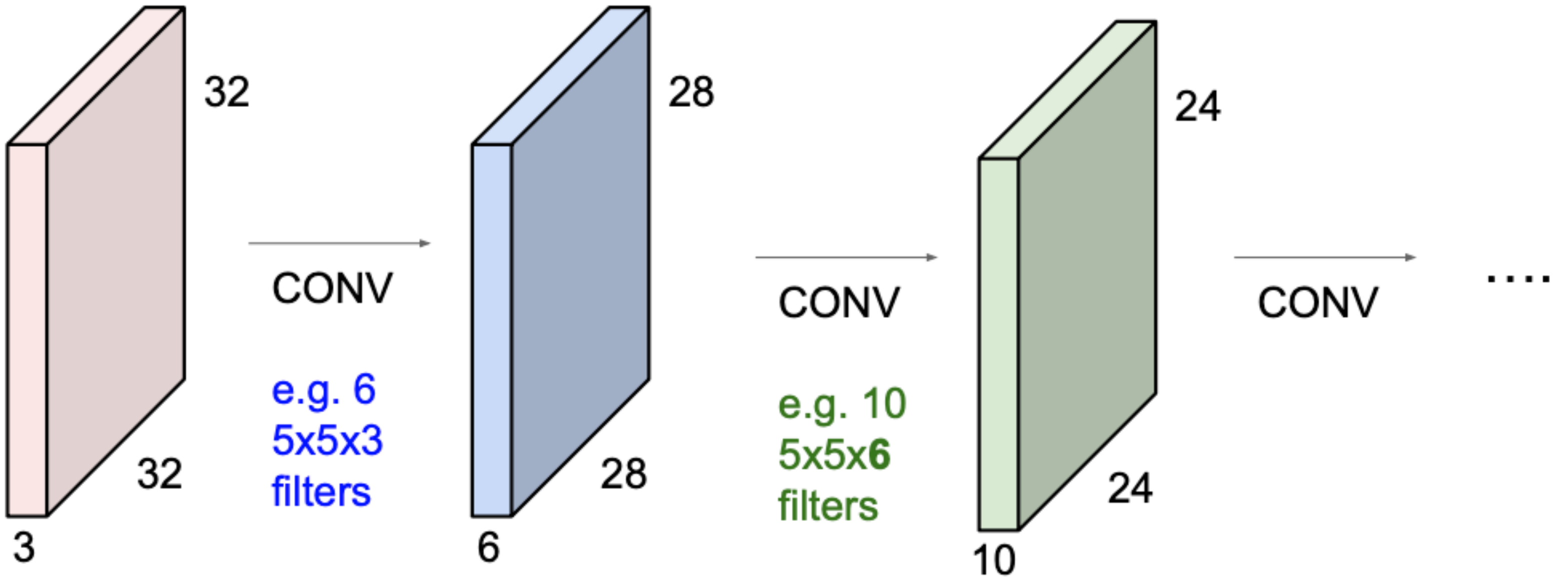
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolutional Networks is a sequence of Convolutional Layers



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

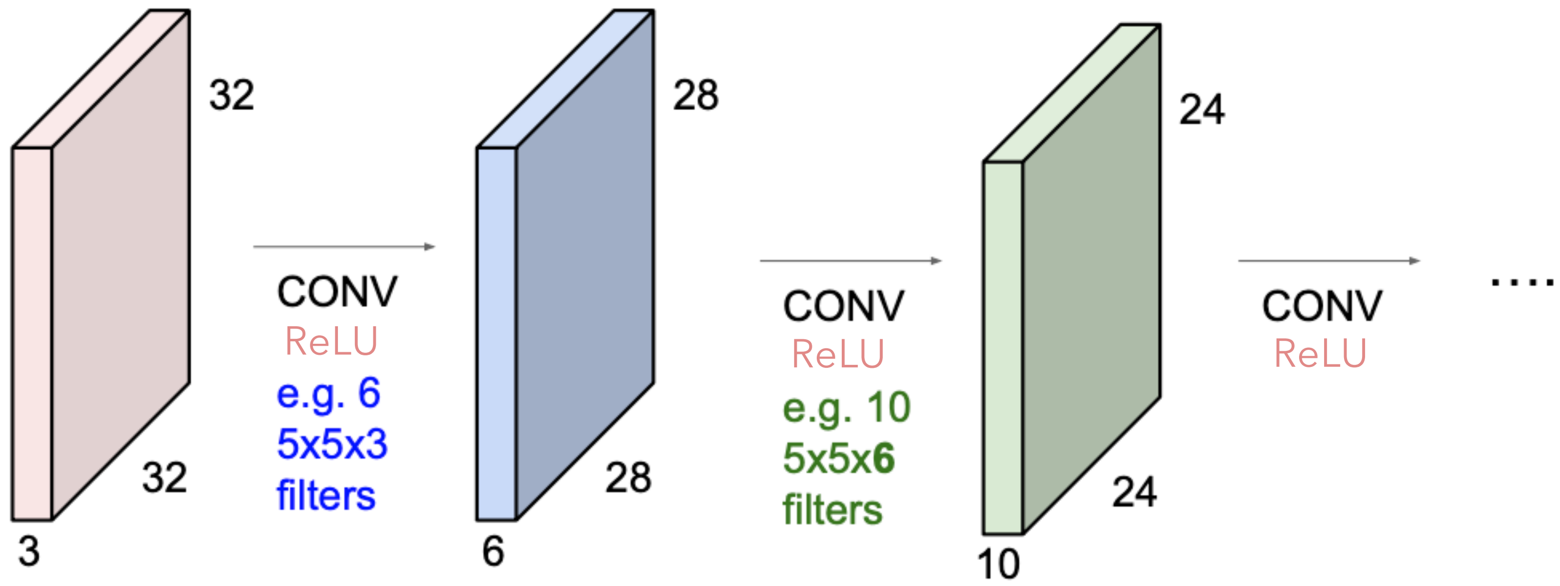
Convolutional Networks is a sequence of Convolutional Layers



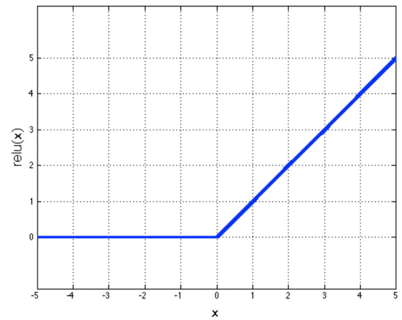
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Convolutional Networks is a sequence of Convolutional Layers

... interspersed with activations functions, making non-linearity possible

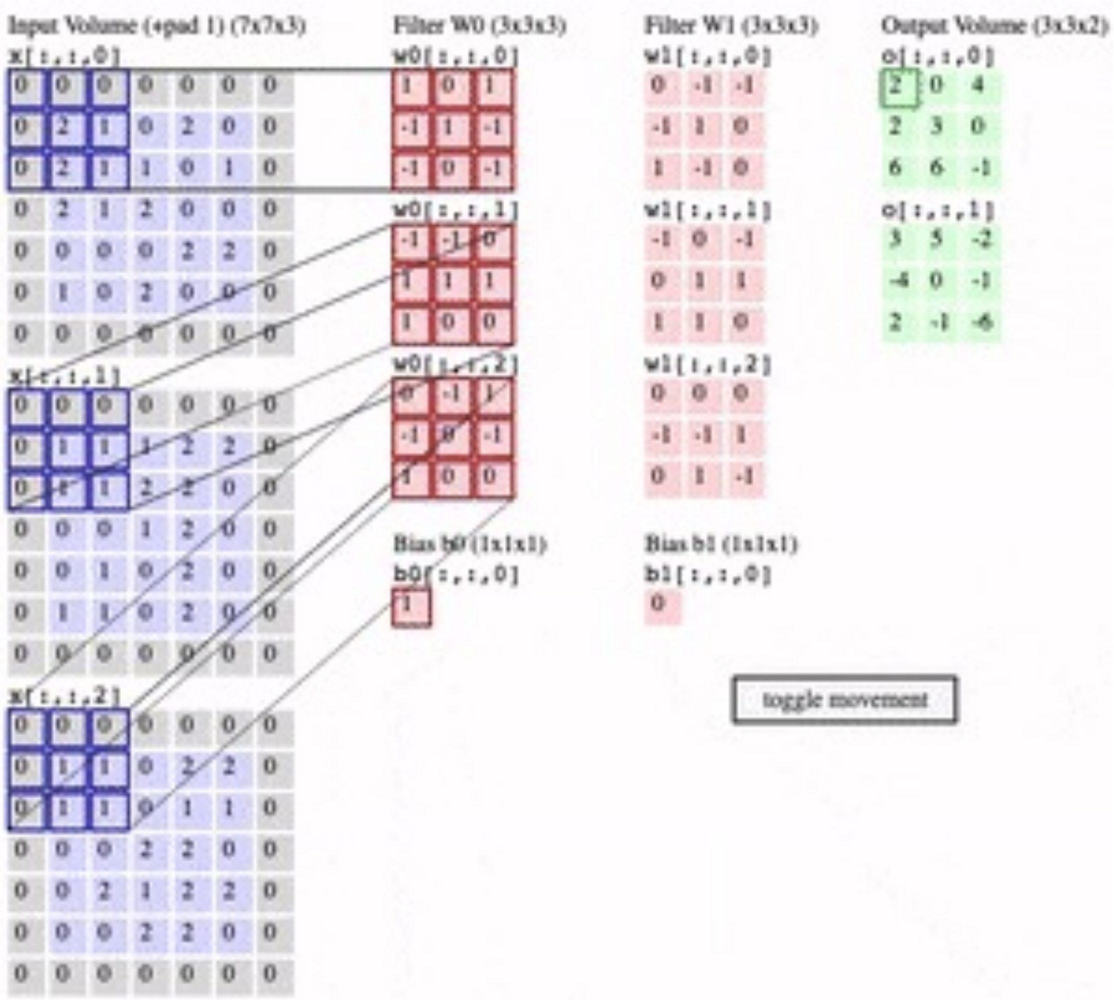


Rectified Linear Unit (ReLU)



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

An example with numbers

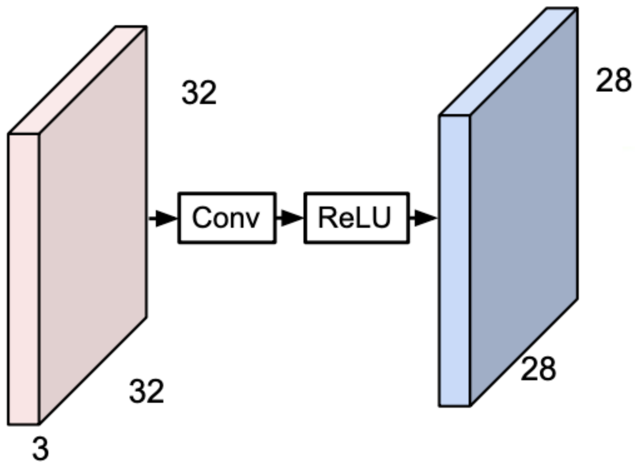


What do the filters learn?

Remember: Linear Classifier → One template/filter per class

“Bank” of whole-image templates

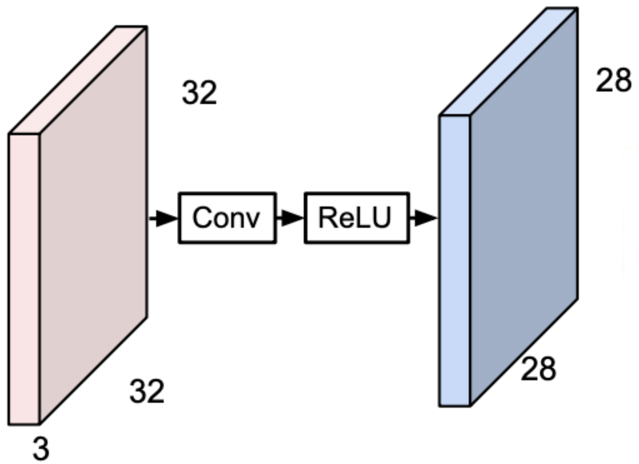
ConvNets: Learn Arbiratry Numbers of filters in multiple layers!



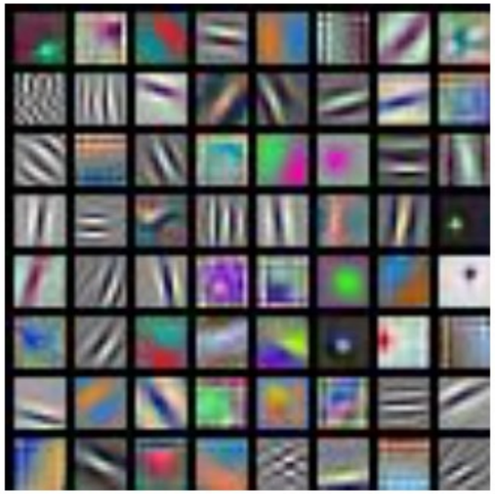
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

What do the filters learn? – Multiple Layers

ConvNets: Learn Arbiratry Numbers of filters in multiple layers!



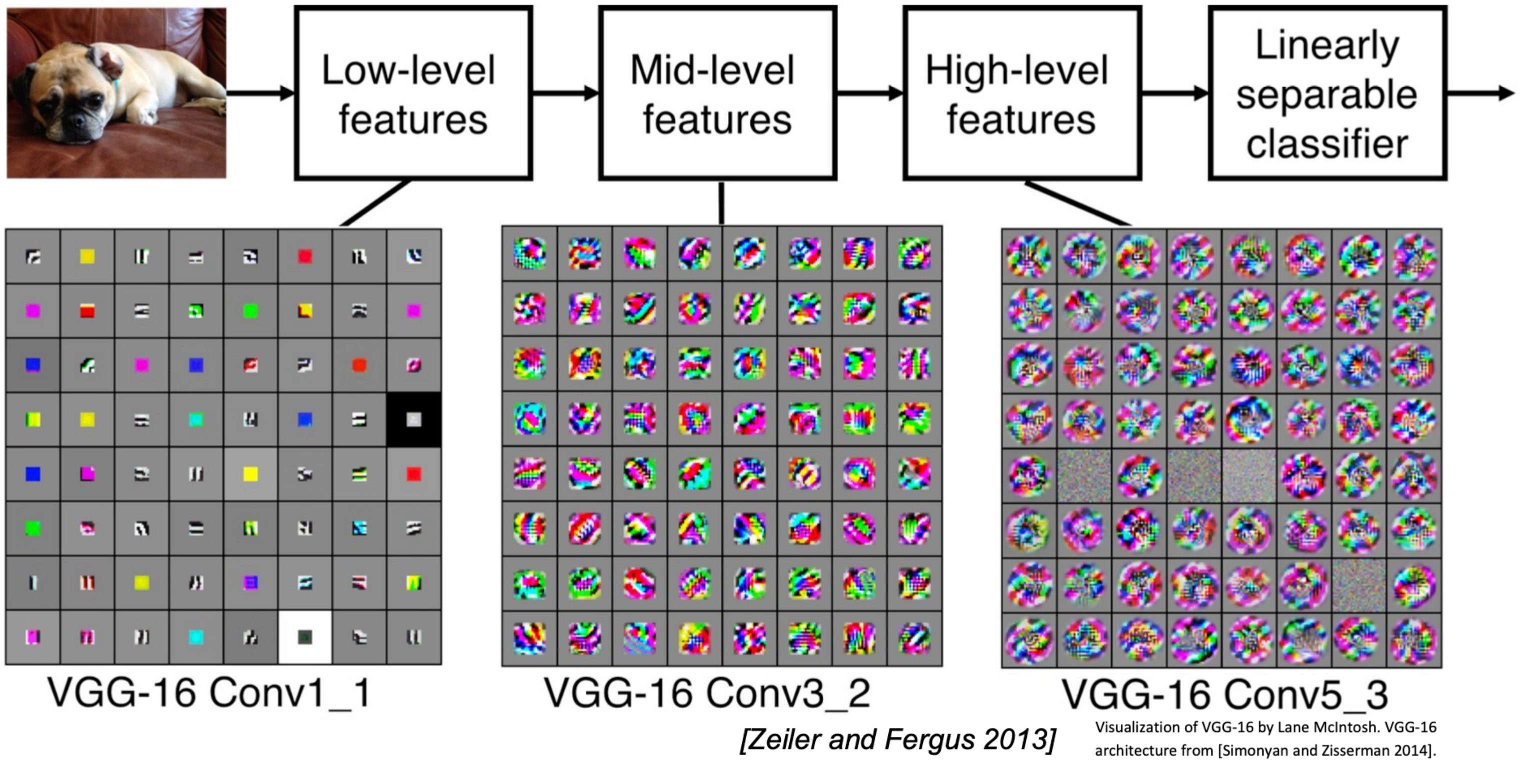
First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)



AlexNet: 64 filters, each 3x11x11

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

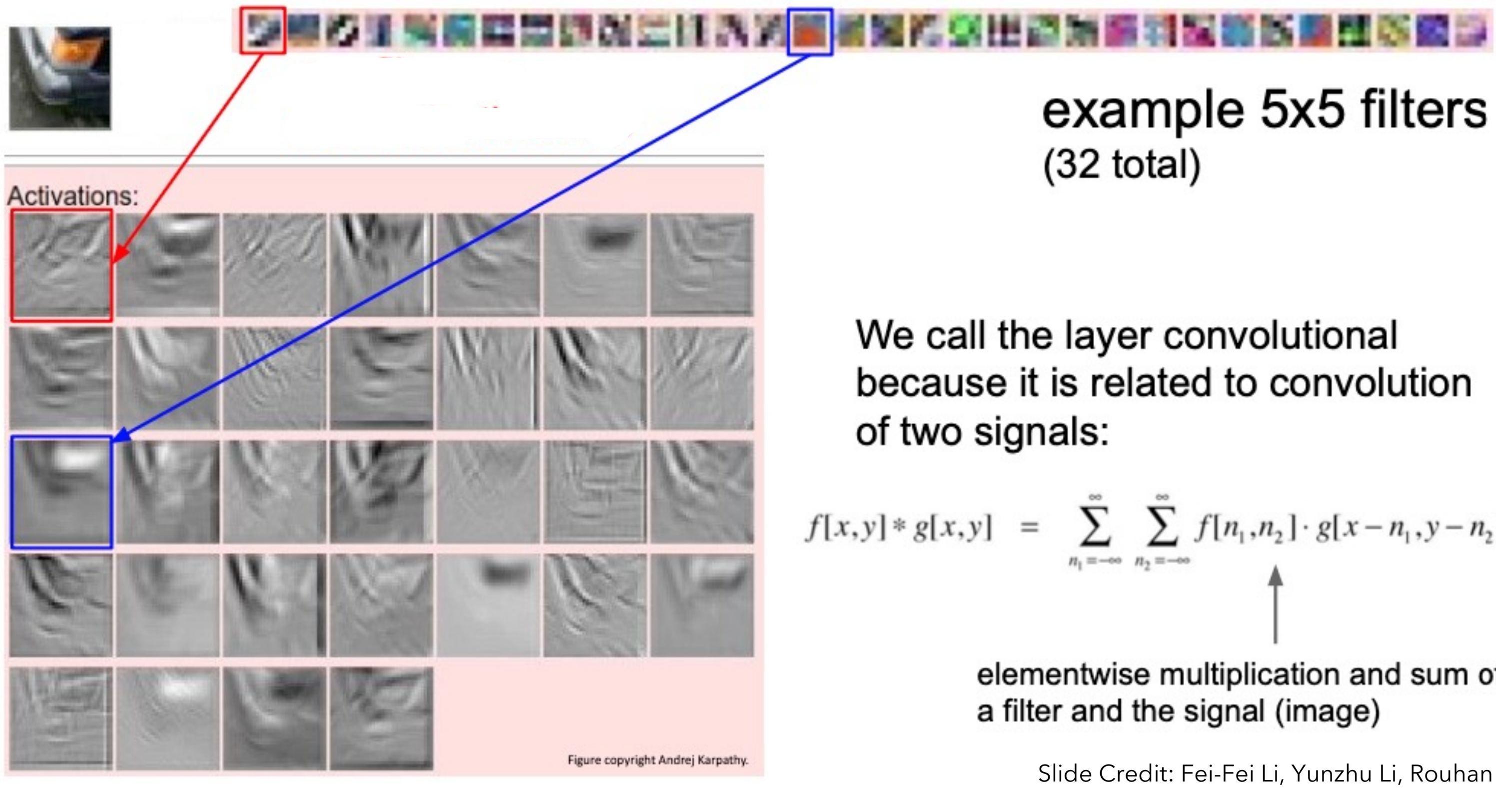
What do the filters learn? – Multiple Layers



Slide Credit: Fei-Fei Li, Justin Johnson

What do the filters learn? – Multiple Layers

Note: One filter → One activation map



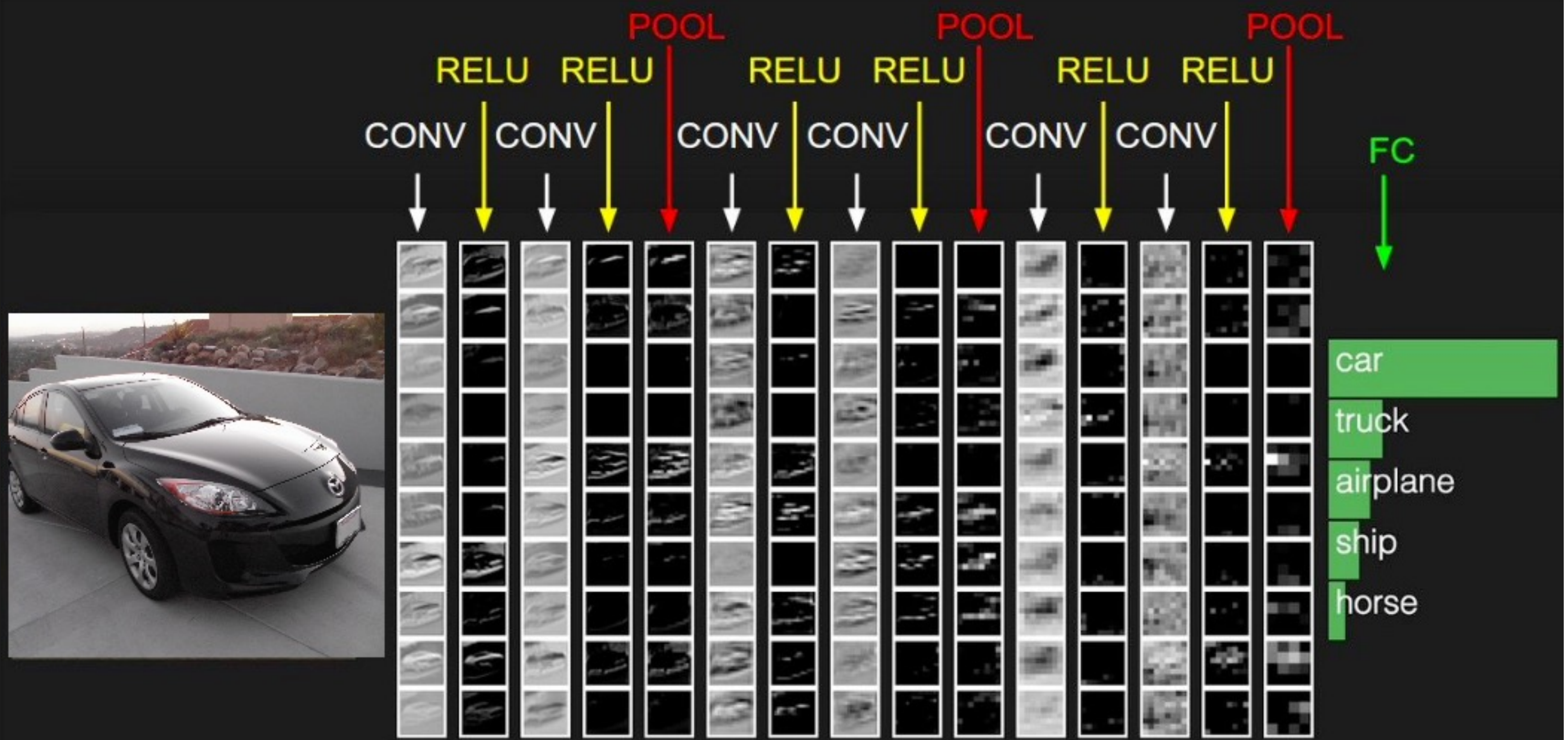
We call the layer convolutional because it is related to convolution of two signals:

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

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

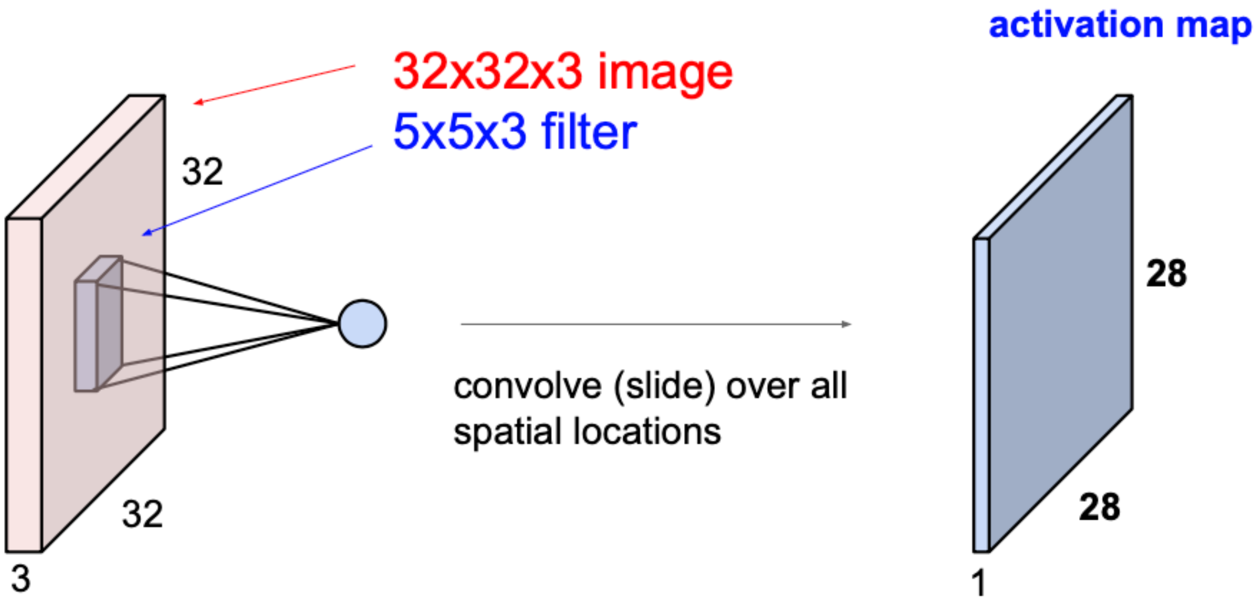
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Network Mechanism Overview



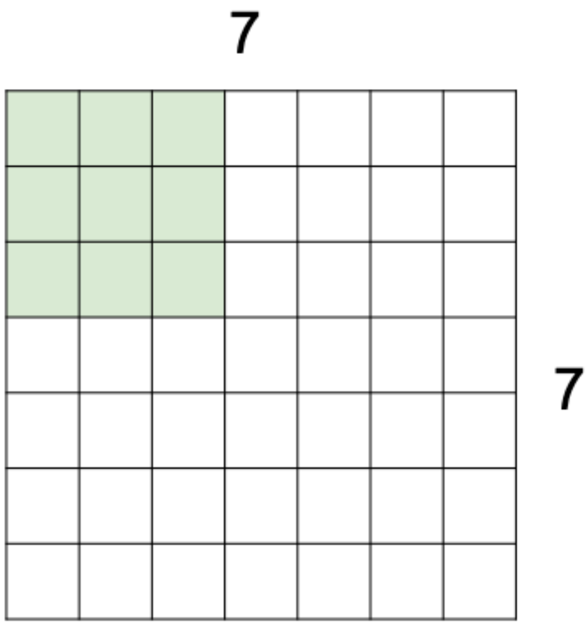
Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: A closer look at spatial dimensions



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

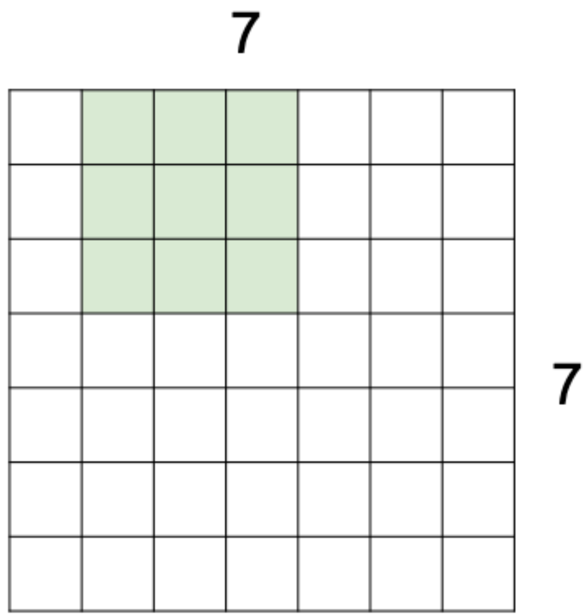
CNN: A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

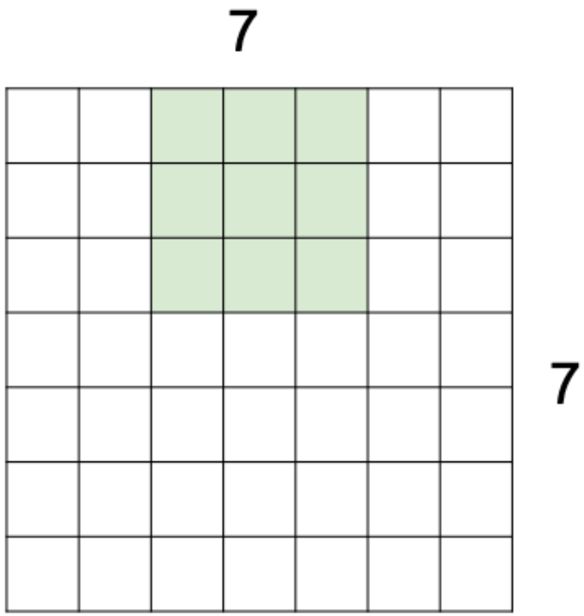
CNN: A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

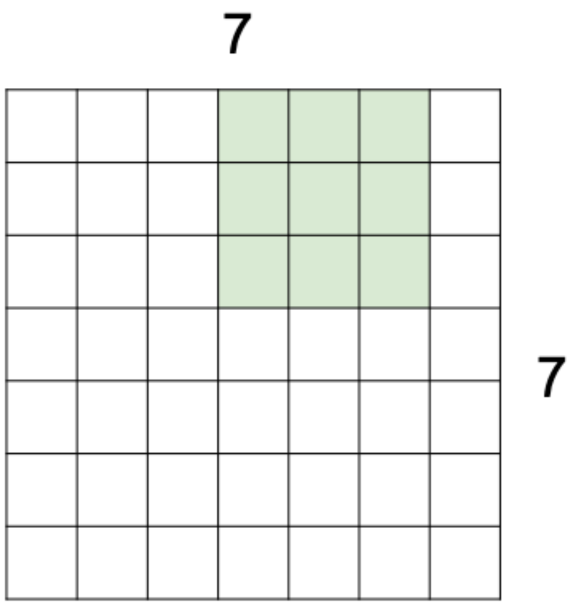
CNN: A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

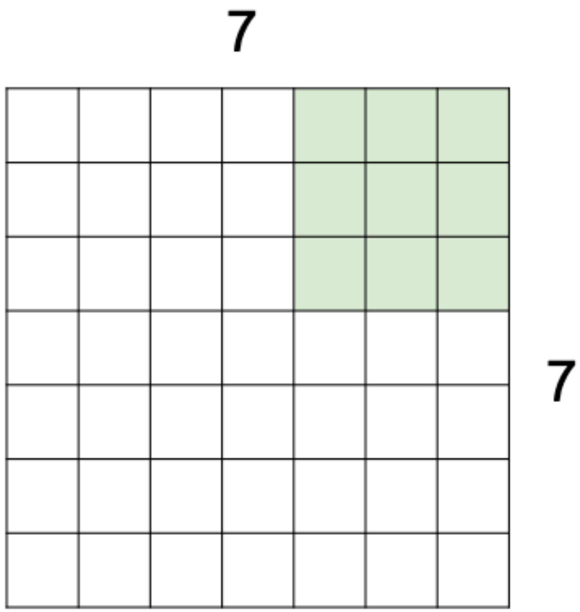
CNN: A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: A closer look at spatial dimensions

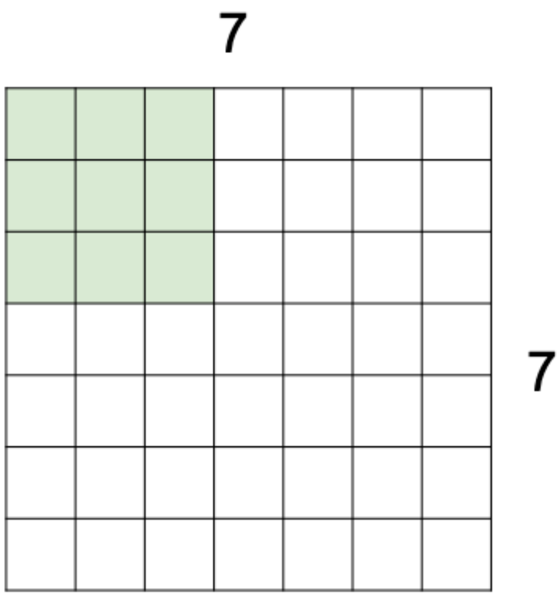


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

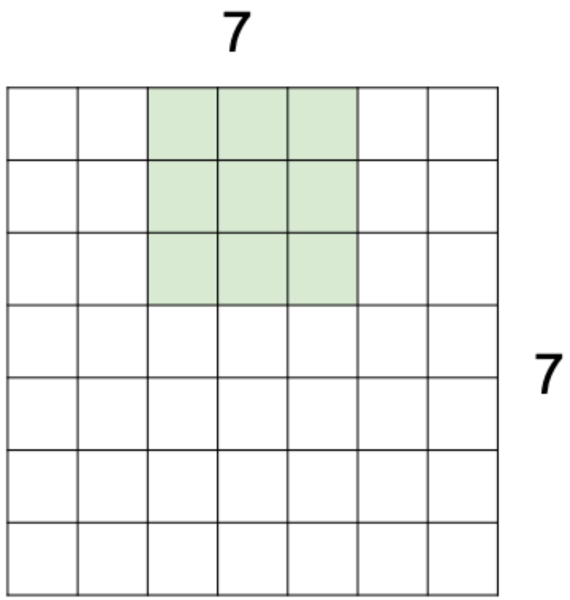
CNN: A closer look at spatial dimensions with *stride*



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

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

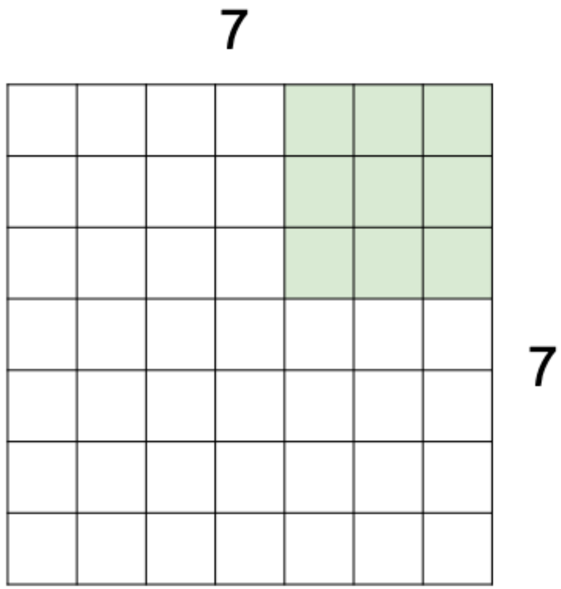
CNN: A closer look at spatial dimensions with *stride*



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

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

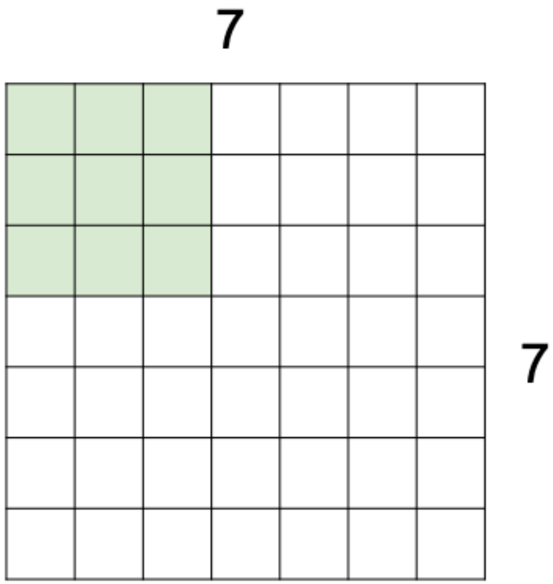
CNN: A closer look at spatial dimensions with *stride*



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

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

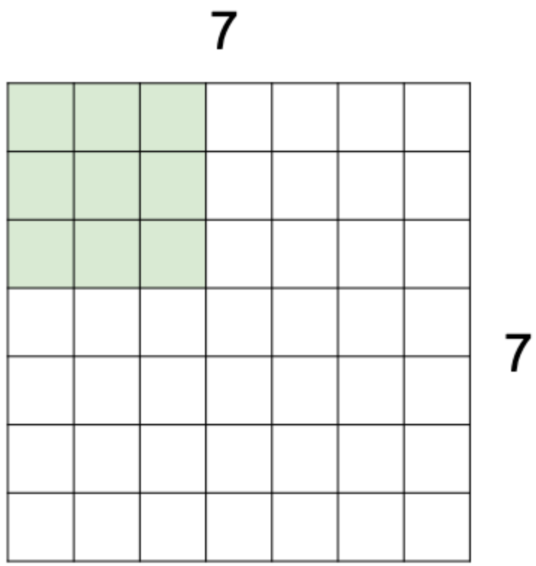
CNN: A closer look at spatial dimensions with *stride*



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

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: A closer look at spatial dimensions with *stride*

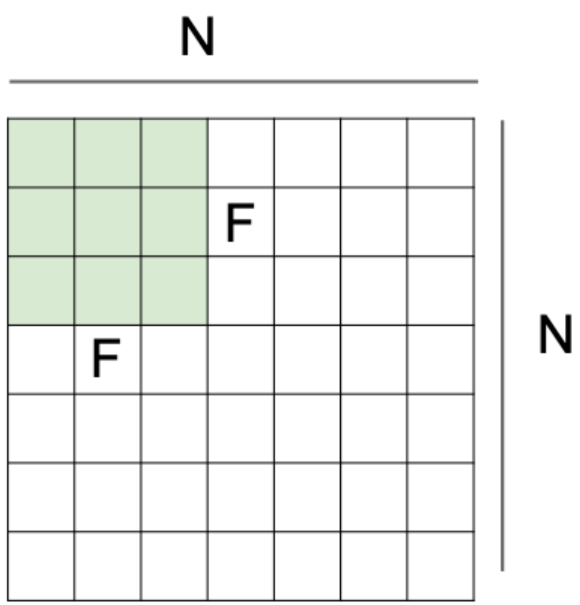


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

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

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: A closer look at spatial dimensions with *stride*



The nature of this mechnism with convolution is the reason behind activation maps gets smaller and smaller

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

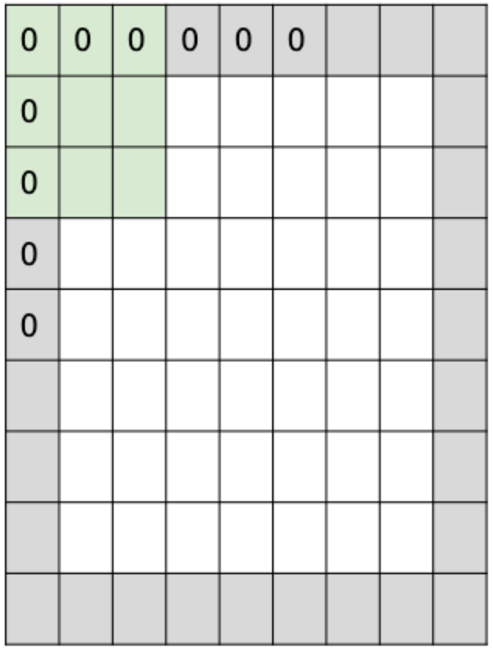
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 😭

Why do we even want to stride?

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: A closer look at spatial dimensions -*padding*

Zero-pad is most common in practice



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

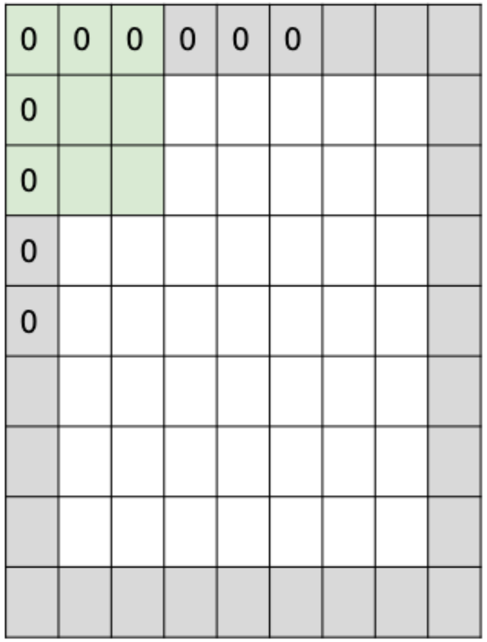
- We pad all around the image to hold the translational invariance.
  The image features is still relative to the original center
- Further as convolution courses the image to shrink, we add padding so the center/ middle layers is not parsed to many more times than the edge layers.

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

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: A closer look at spatial dimensions -*padding*

Zero-pad is most common in practice

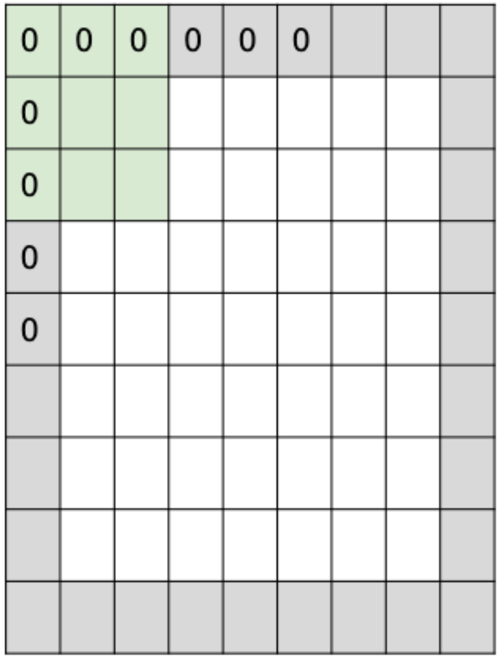


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

CNN: A closer look at spatial dimensions - *padding*

Zero-pad is
most common
in practice



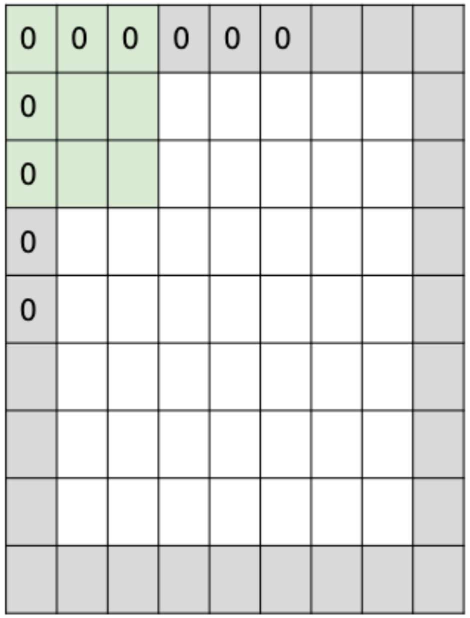
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

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

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: A closer look at spatial dimensions -*padding*



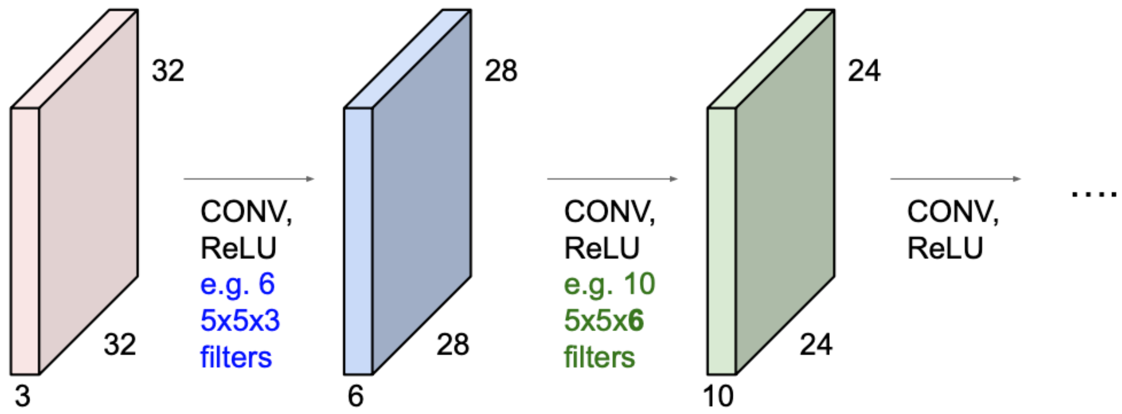
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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 Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: The Volume Shrinks

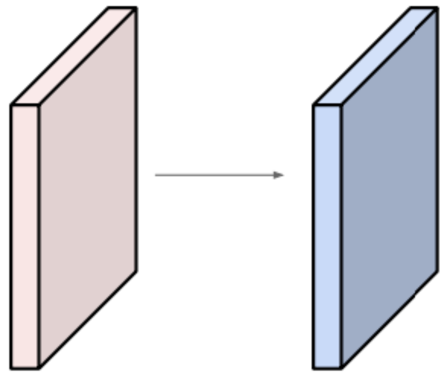
- A 32x32 input convoled repeatedly with a 5x5 filters shrinks the volume spatially
- 32 → 28 → 24 …
- Shrinking to fast: Not good
  - But shrinking is computationally a nice feature
  - Trade off



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: The Volume Shrinks, *Example*

Examples time:



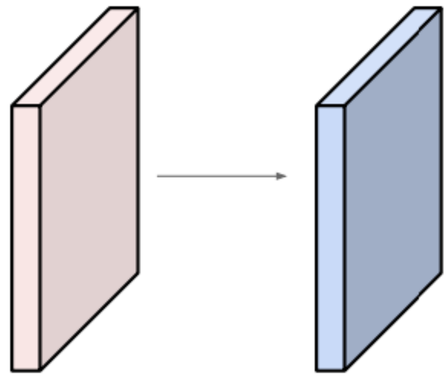
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: The Volume Shrinks, *Example*

Examples time:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

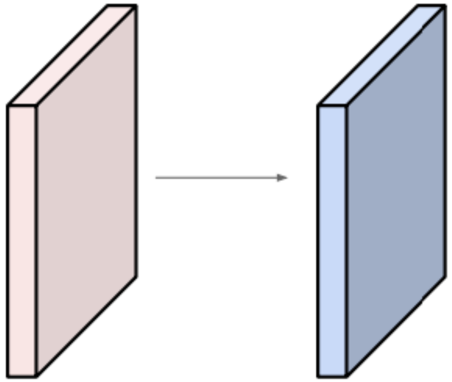
Output volume size:

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

32x32x10

CNN: The Volume Shrinks, *Example*

Examples time:



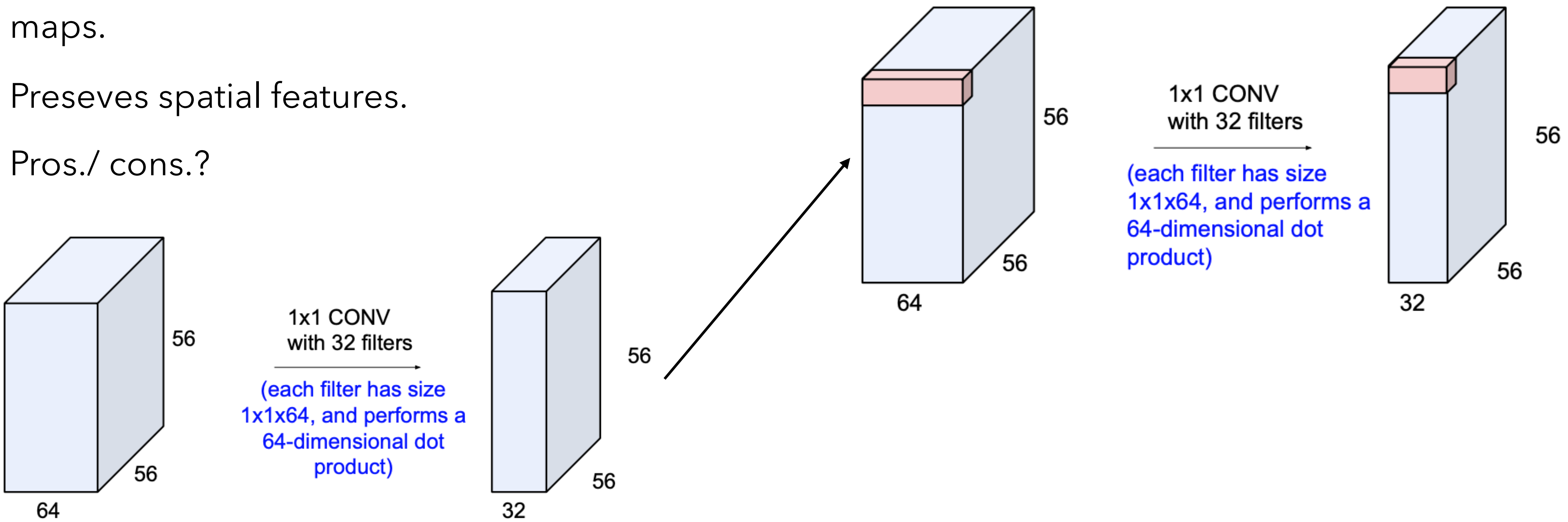
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CNN: The Volume Shrinks: 1x1 Convolution makes sense

- Collabs Information Across channels/ feature maps.
- Preseves spatial features.
- Pros./ cons.?



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

CONV Layer in Pytorch

- Conv Layers needs 4 hyperparamters
  - Number of filter ***k***
  - The filter size ***F***
  - The stride ***S***
  - The zero padding ***P***

SpatialConvolution

```
module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH])
```

Applies a 2D convolution over an input image composed of several input planes. The `input` tensor in `forward(input)` is expected to be a 3D tensor (`nInputPlane x height x width`).

The parameters are the following:

- `nInputPlane` : The number of expected input planes in the image given into `forward()` .
- `nOutputPlane` : The number of output planes the convolution layer will produce.
- `kW` : The kernel width of the convolution
- `kH` : The kernel height of the convolution
- `dW` : The step of the convolution in the width dimension. Default is `1` .
- `dH` : The step of the convolution in the height dimension. Default is `1` .
- `padW` : The additional zeros added per width to the input planes. Default is `0` , a good number is `(kW-1)/2` .
- `padH` : The additional zeros added per height to the input planes. Default is `padW` , a good number is `(kH-1)/2` .

Note that depending of the size of your kernel, several (of the last) columns or rows of the input image might be lost. It is up to the user to add proper padding in images.

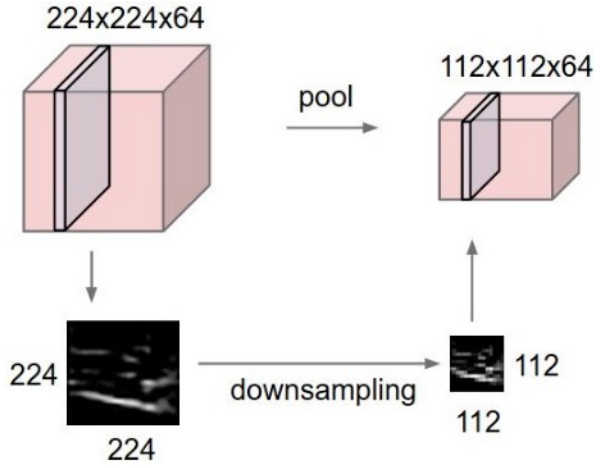
If the input image is a 3D tensor `nInputPlane x height x width` , the output image size will be `nOutputPlane x oheight x owidth` where

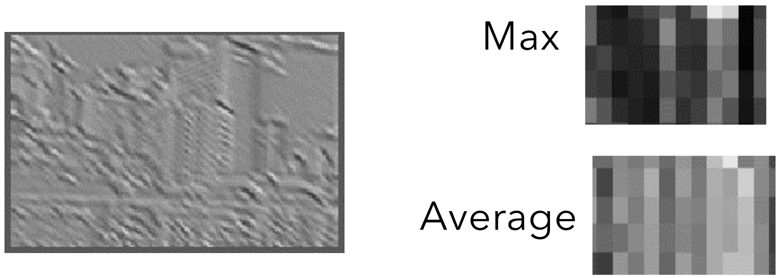
```
owidth  = floor((width  + 2*padW - kW) / dW + 1)
oheight = floor((height + 2*padH - kH) / dH + 1)
```

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Pooling Layer

- Makes the representation smaller and more managable
- Invariance to small transformations
- Operates over each activation map indepently

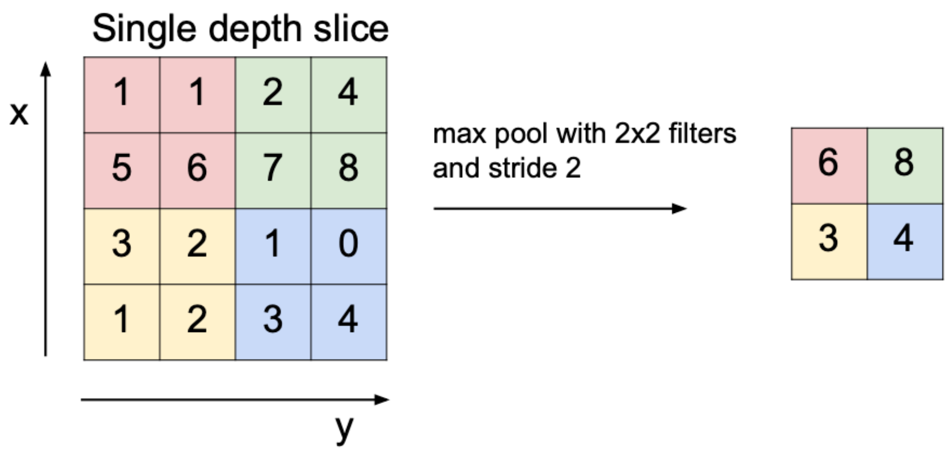




Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Pooling Layer

- Max Pool
- Min Pool
- Average Pool
- Introduces the spatial invariance (as long we pad all around input x, y dim)



Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

Pooling Layer

- Assume input: $W_1 \times H_1 \times C$
- Conv Layer needs 2 hyperparamters
  - The spatial extend (filter) ***F***
  - The stride ***S***

  - This will produce and output : $W_2 \times H_2 \times C$
    - $W_2 = (W_1 - F) / S + 1$
    - $H_2 = (H_1 - F) / S + 1$
    - Number of parameters: **0**

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

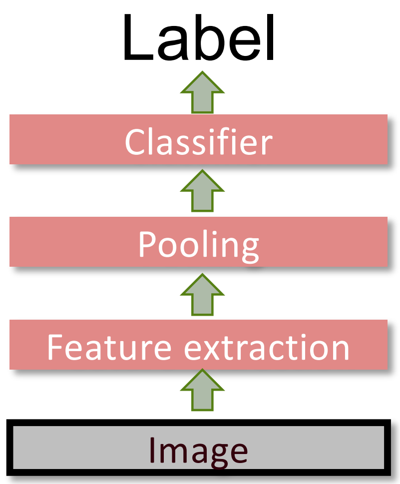
Pooling Layer vs. Stride

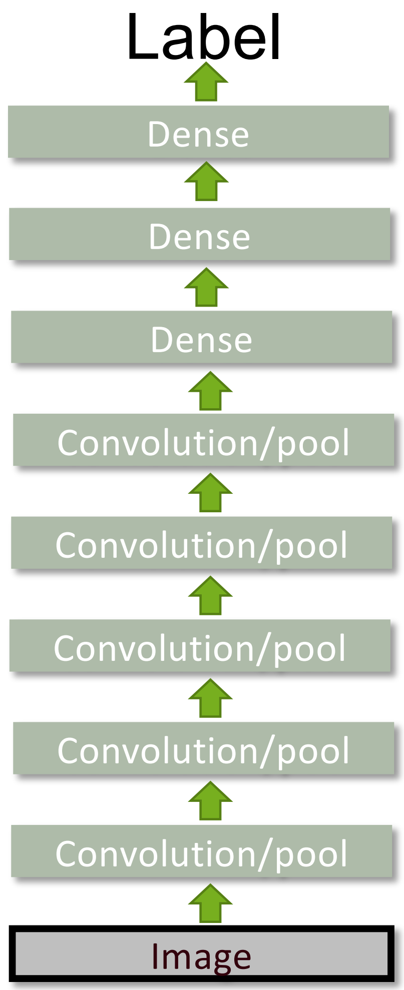
- The pooling layer reduces the spatial dimensions helping the computional overhead *and* also forces the network to destill the “correct” information as weighted with respect to the pool-function used (min, max, avg)
- The strides are responsible for regulating the features that could be missed while flattening the image.
  - How coarse / dominant do we accept the features to be.
- Overall they control the networks sensitivity to the features in the image.

Slide Credit: Fei-Fei Li, Yunzhu Li, Rouhan Gao

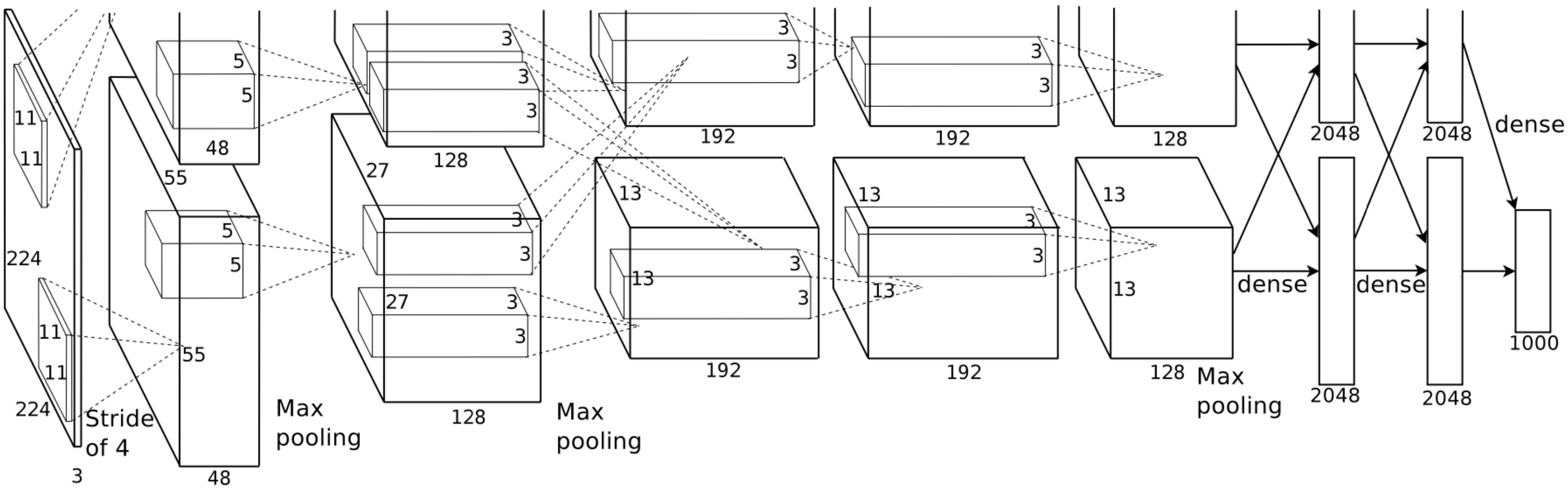
Engineered vs. Learned Features

Convolutional filters are trained in a supervised manner by back-propagating classification error





A Common Architecture: *AlexNet*



link

A Common Architecture: *ResNet*

- ResNet: Residual Networks

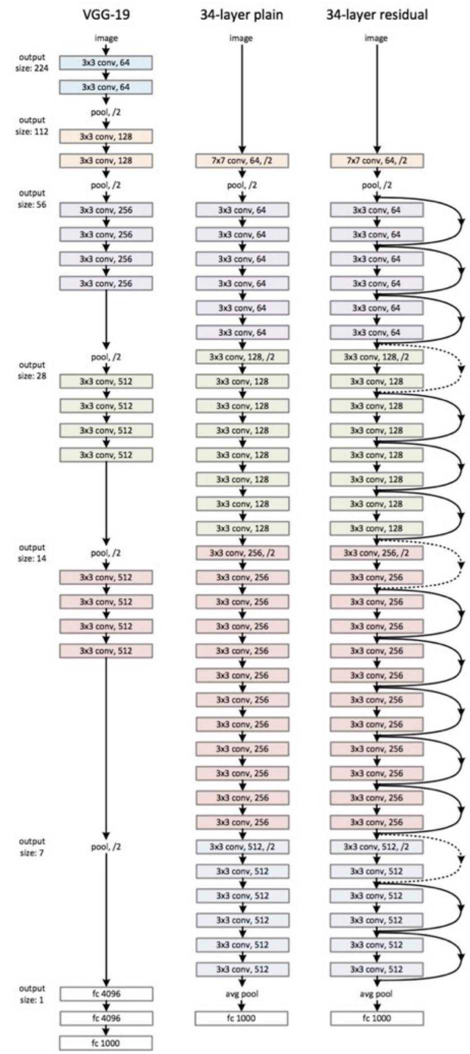


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Beyond Classification

Detection

Segmentation

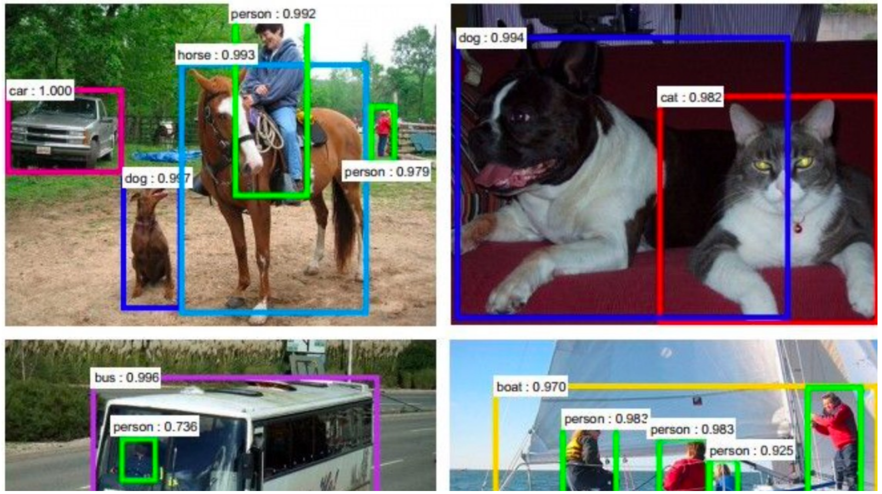
Regression

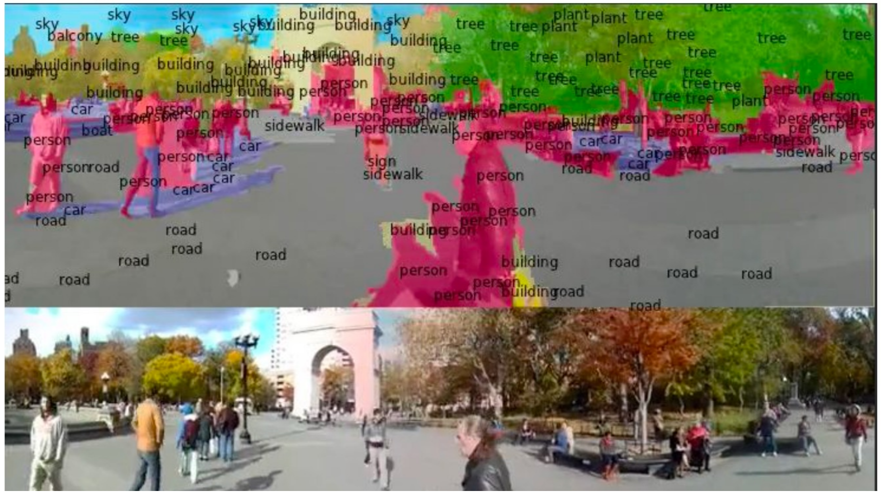
Pose estimation

Matching patches

Synthesis

and many more…





[Farabet et al., 2012]

Deep Neural Networks: CNNs