CS60010: Deep Learning

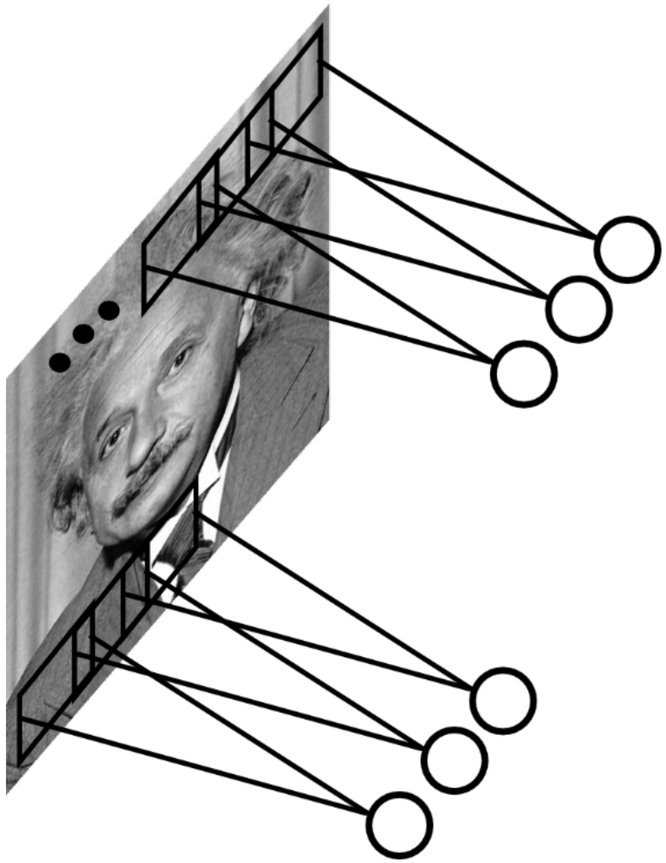
Spring 2023

Sudeshna Sarkar

CNN Part 2

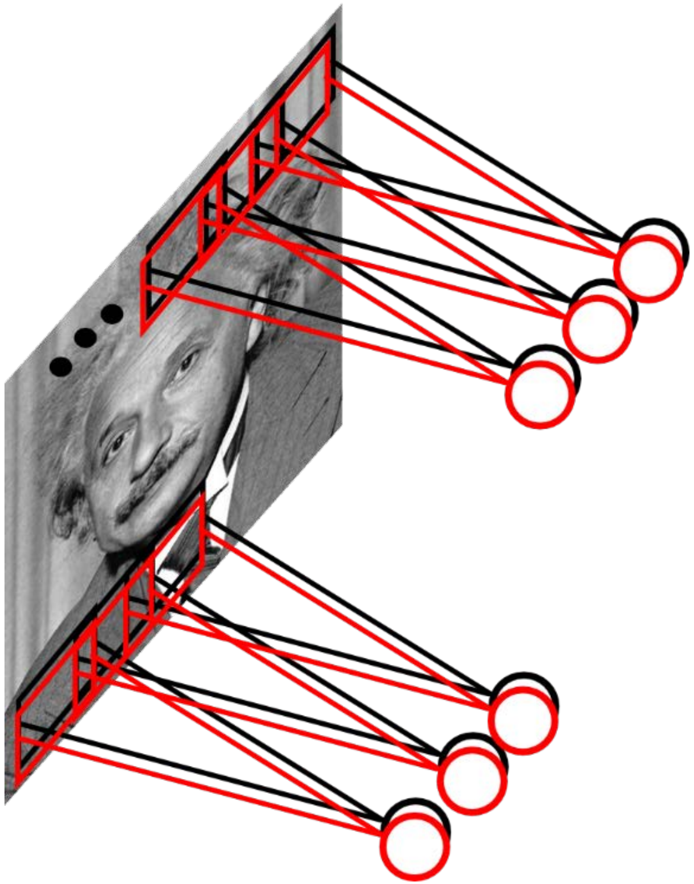
8 Feb 2023

Convolutional Layer



Share the same parameters across different locations (assuming input is stationary): Convolutions with learned kernels

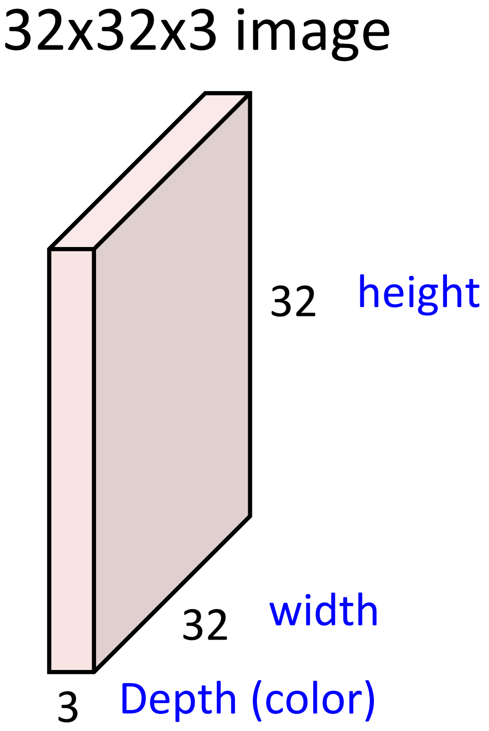
Convolutional Layer



Learn multiple filters.

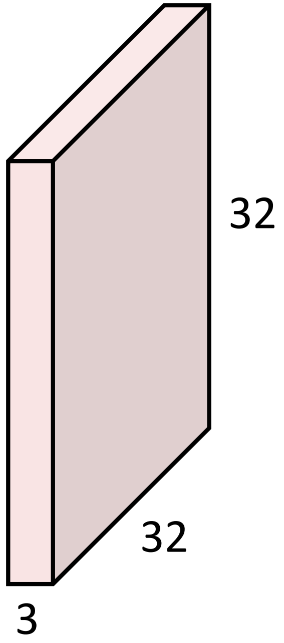
E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Convolution Layer

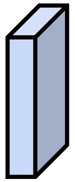


Convolution Layer

32x32x3 image



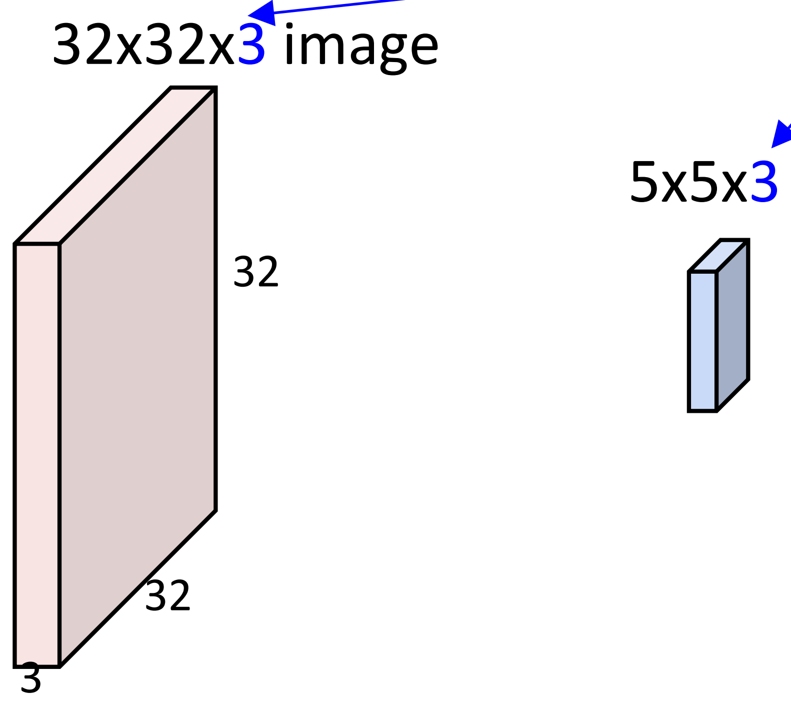
5x5x3 filter



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

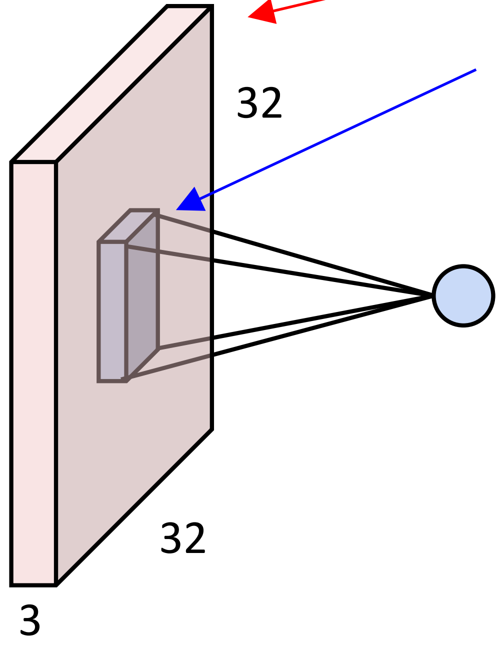
Convolution Layer

Filters always extend to the full depth of the input volume



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

Convolution Layer



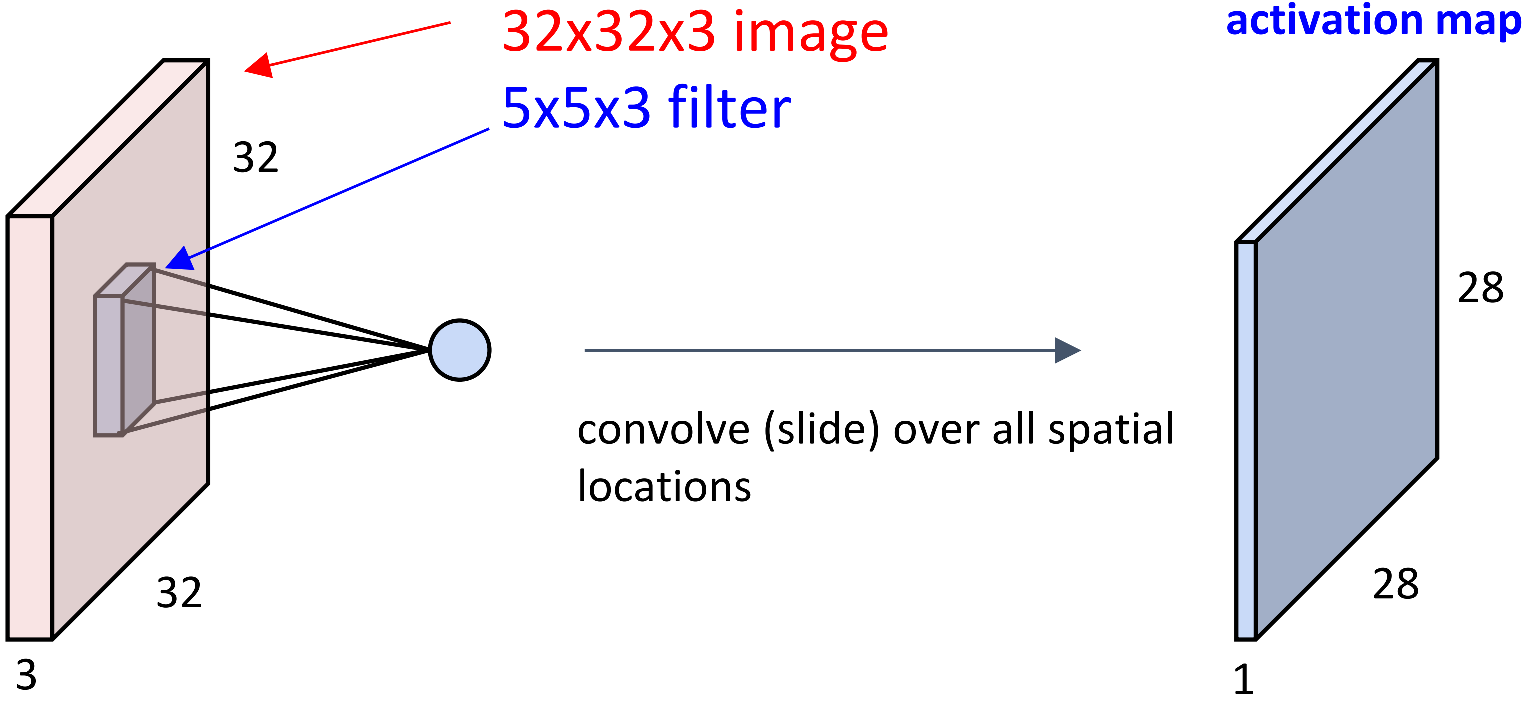
32x32x3 image

5x5x3 filter $\boxed{w}$

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

$$\boxed{w^T x + b}$$

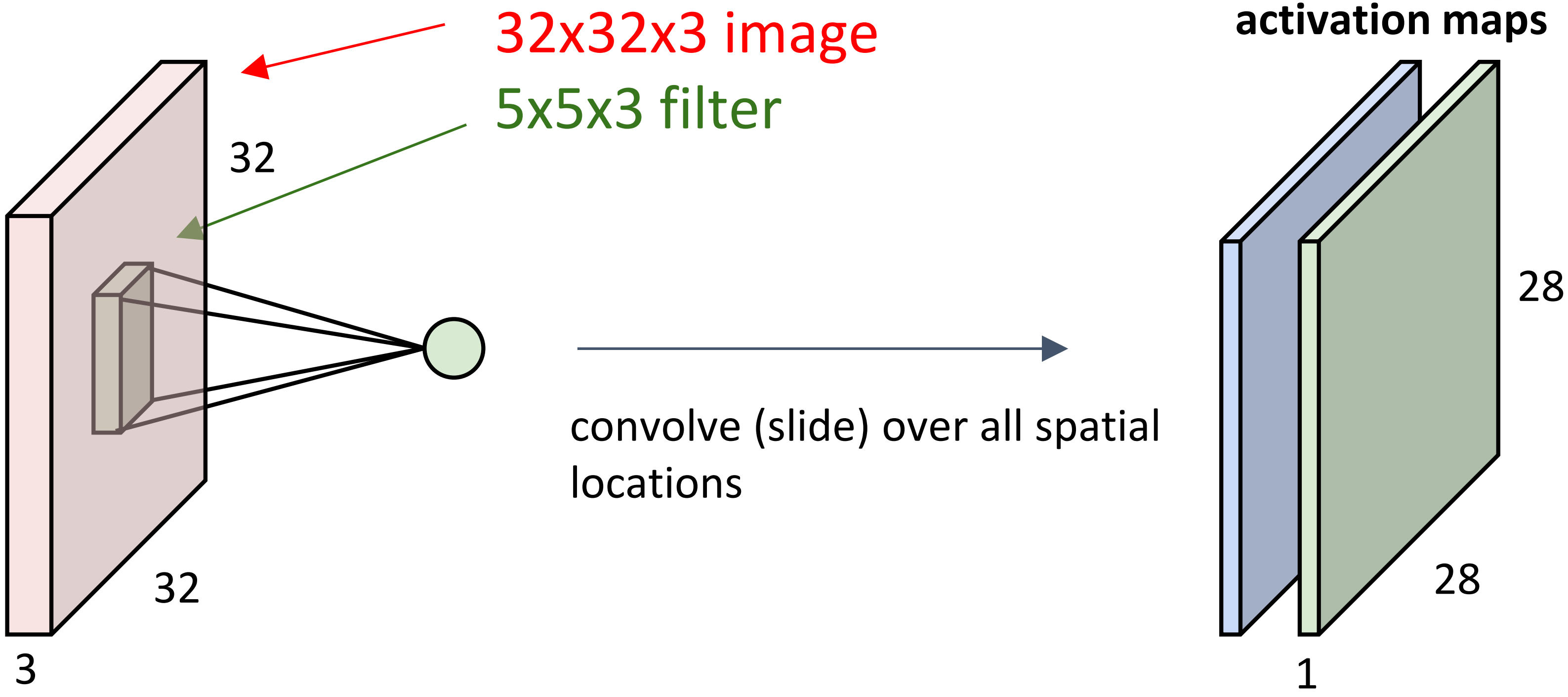
Convolution Layer



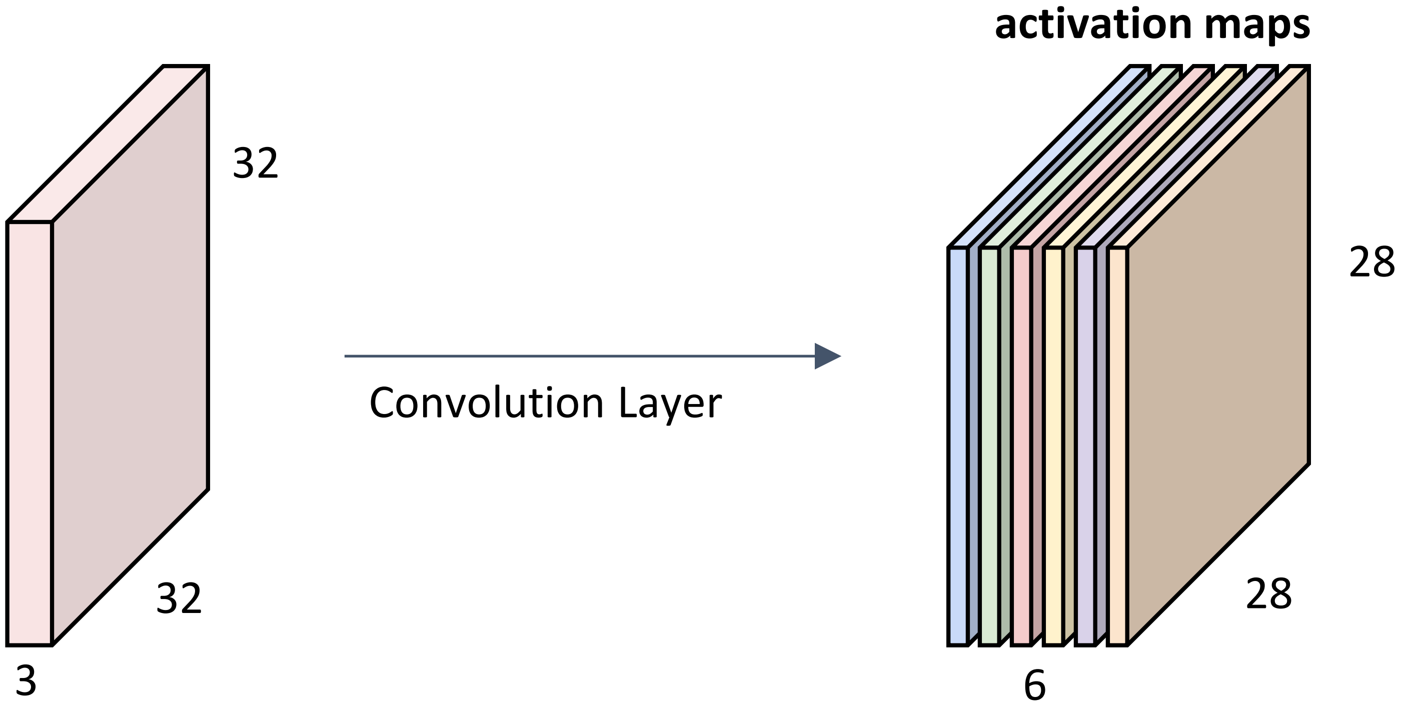
convolve (slide) over all spatial locations

Convolution Layer

consider a second, green filter



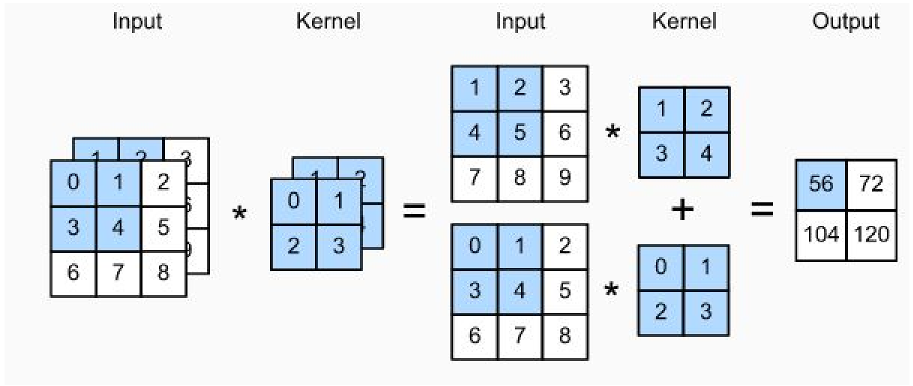
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

Multiple Input Channels

- When the input data contain multiple channels, we need to construct a convolution kernel with the same number of input channels as the input data.
- Number of input channels : c_i
- If the convolution kernel window shape is $k_h \times k_w$
- Shape of convolution kernel: $k_h \times k_w \times c_i$



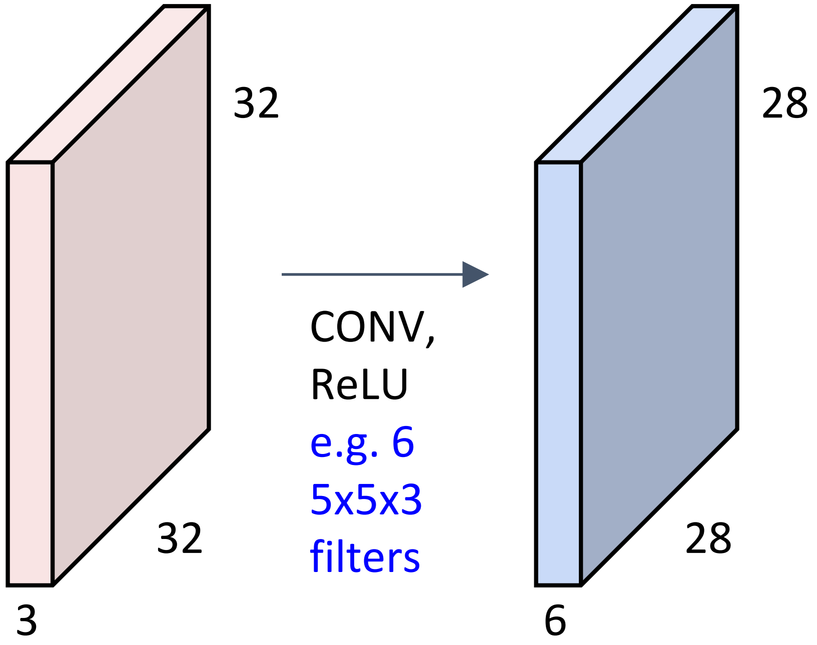
“Tensors”

Because there are multiple channels in each data block, convolutional filters are normally specified by 4D arrays. Common dimensions are:

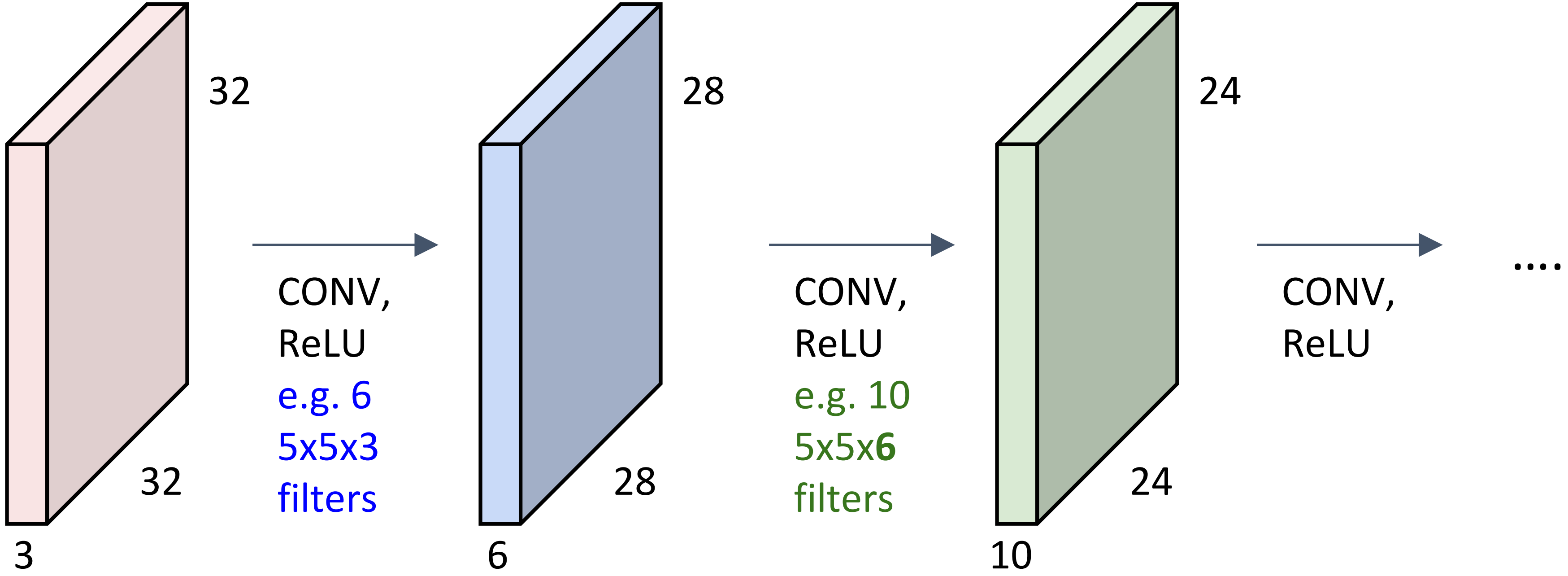
$$C_{out} \times C_{in} \times F_H \times F_W$$

C_{out} = number of output channels
C_{in} = number of input channels
F_H = filter height
F_W = filter width

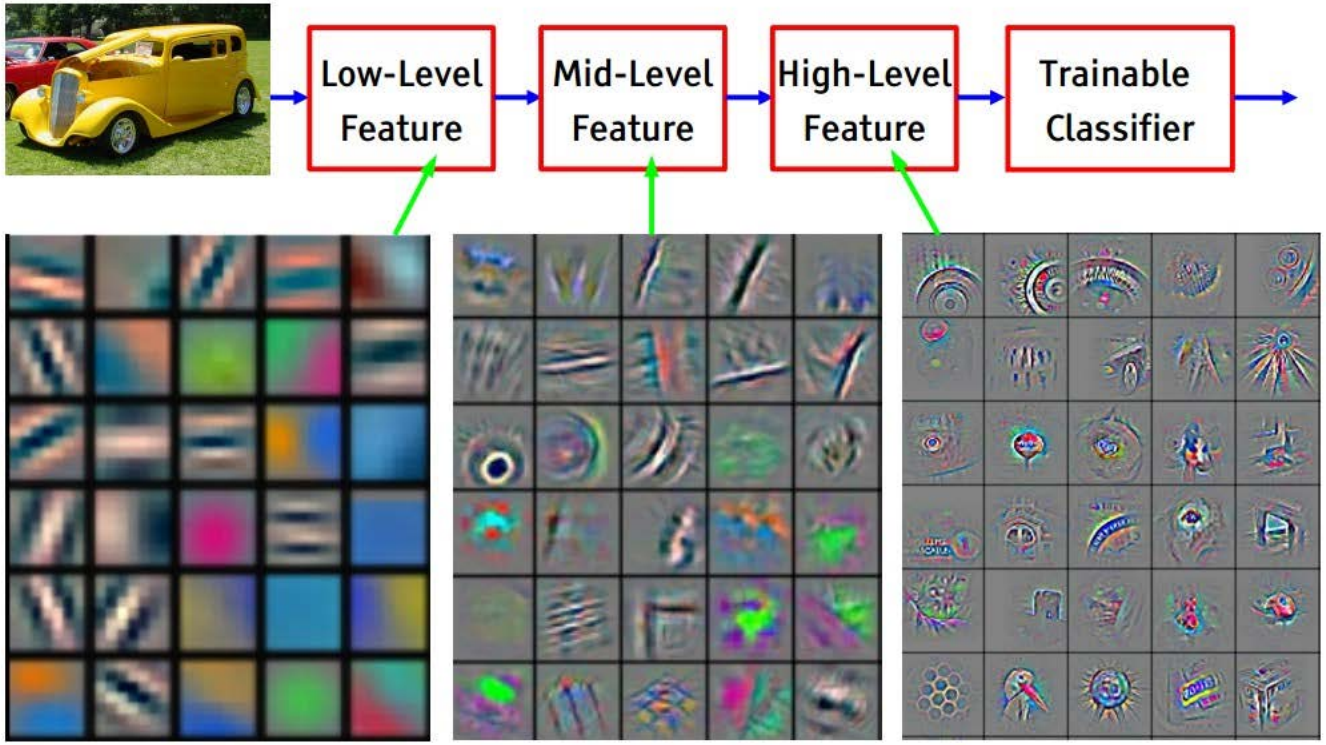
Preview: ConvNet is a sequence of Convolution Layers, interspersed with non-linear activation functions



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



Recall



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

Convolution

Filter

0	1	2
2	2	0
0	1	2

Image

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

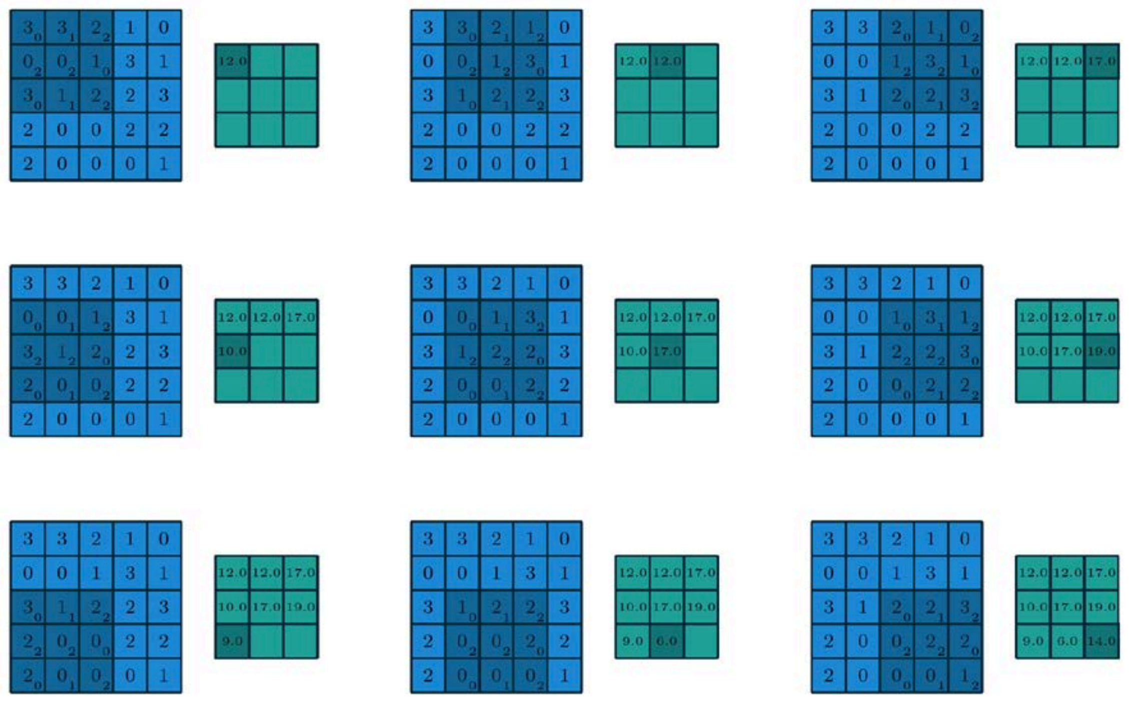


Image courtesy:Vincent Dumoulin

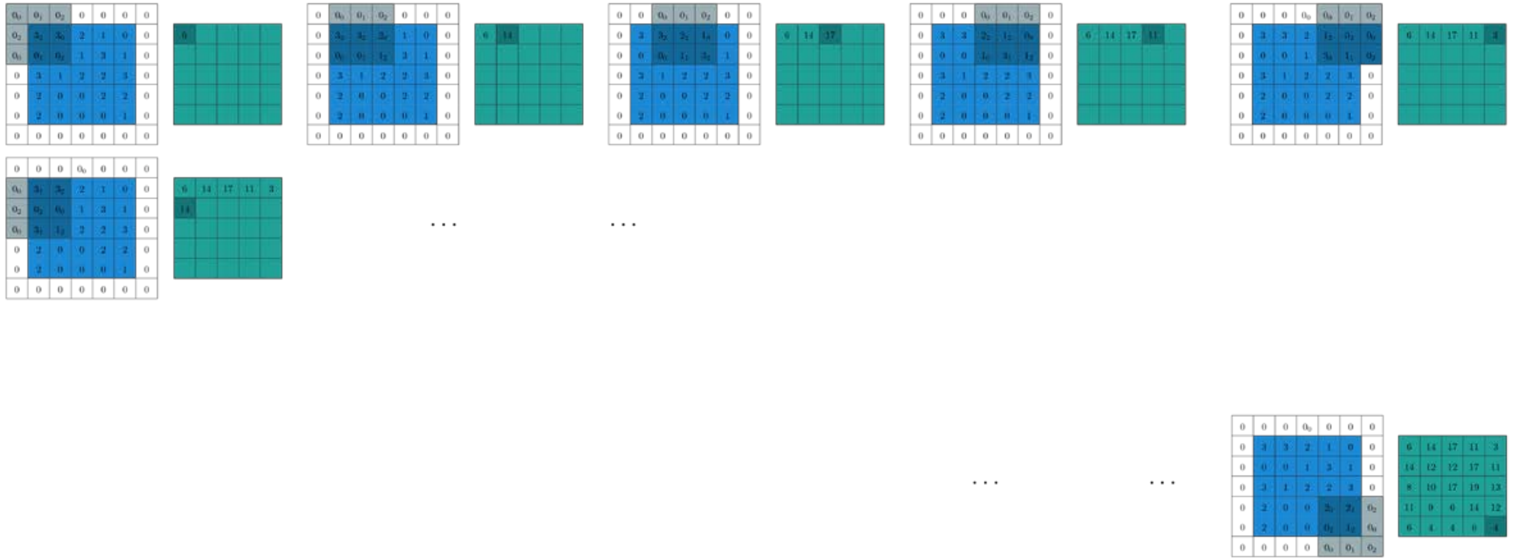
Convolution with Zero Padding

Filter

0	1	2
2	2	0
0	1	2

Image

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



Convolution with Strides

Convolving a 3x3 kernel over a 5x5 input using 2x2 strides

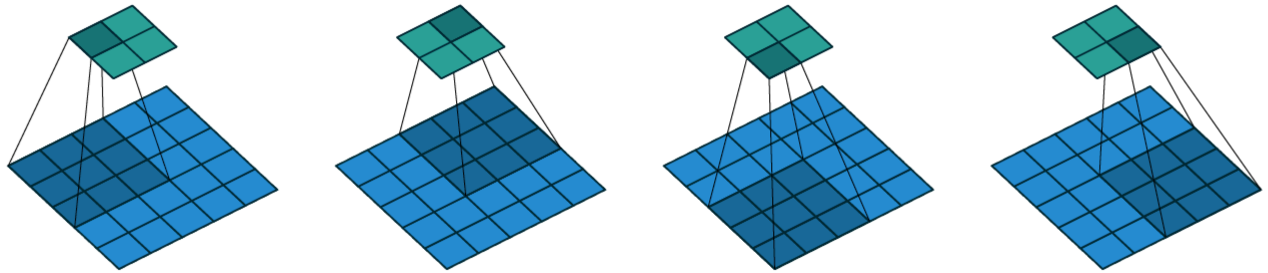


Image courtesy:Vincent Dumoulin

Convolution with Strides and Zero Padding

Convolving a 3x3 kernel over a 5 x5 input using 1 x1 strides and 1 zero-padding

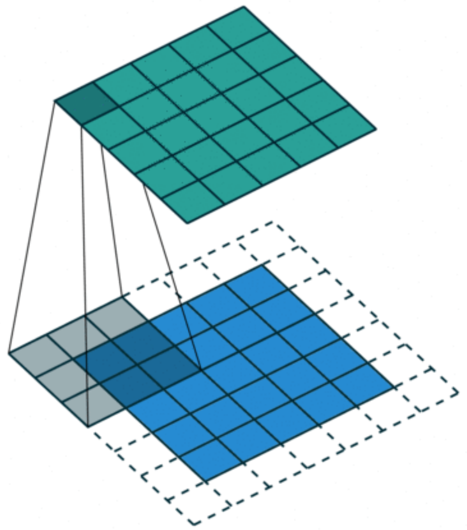
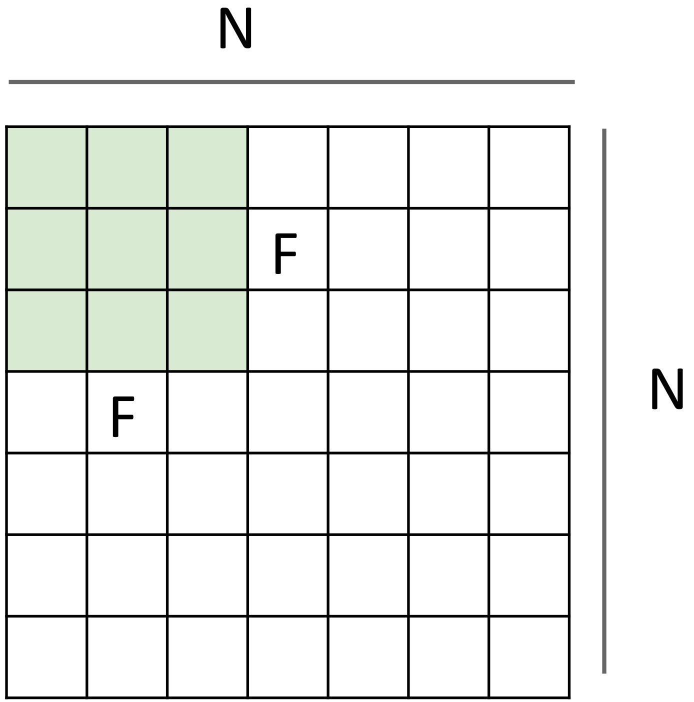


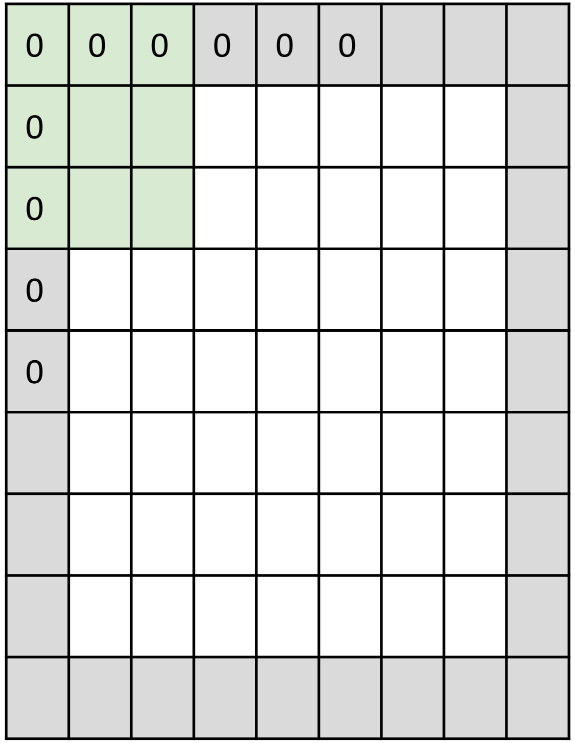
Image courtesy:Vincent Dumoulin



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 :\

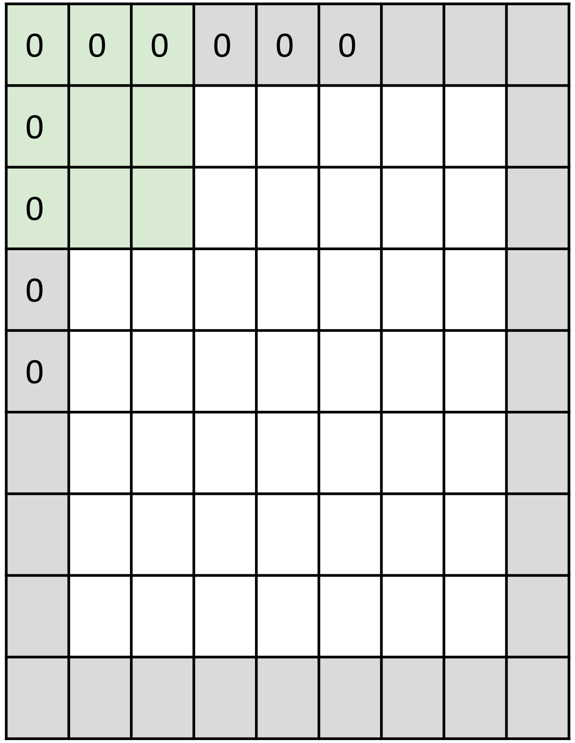
In practice: Common to zero pad the border



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

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

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!
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

Padding

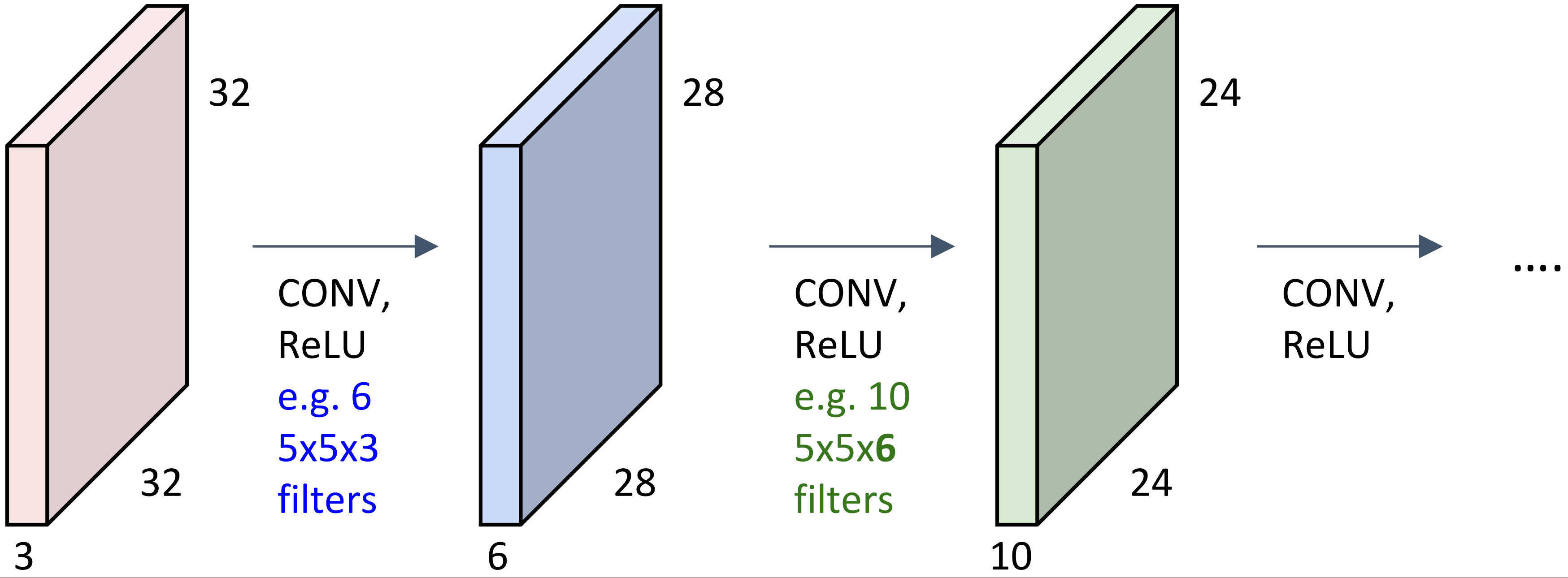
- We tend to lose pixels on the perimeter of our image.
- Add extra pixels of filler around the boundary of the input image, thus increasing the effective size of the image.

$$D_{out} = (N + 2P - F)/\mathrm{stride} + 1$$

$$D_{out} = \lfloor (N + 2P - F)/\mathrm{stride} \rfloor + 1$$

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.



When the Filter doesn’t fit:

Our new formula with padding on both sides is:

$$D_{out} = (N + 2P - F)/\text{stride} + 1$$

(the effective image size is now $N + 2P$).
This means that the number of strided steps we can take may not be an integer for $S > 1$.
But we can still take the next-smallest integer number of steps.
In practice almost all deep learning toolkits simply round down to the nearest integer in this case (disregard the cs231n notes on this point). i.e.

$$D_{out} = \lfloor (N + 2P - F)/\text{stride} \rfloor + 1$$

Examples time:

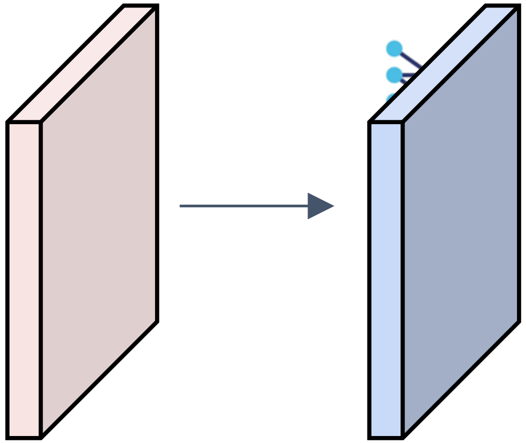
Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

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

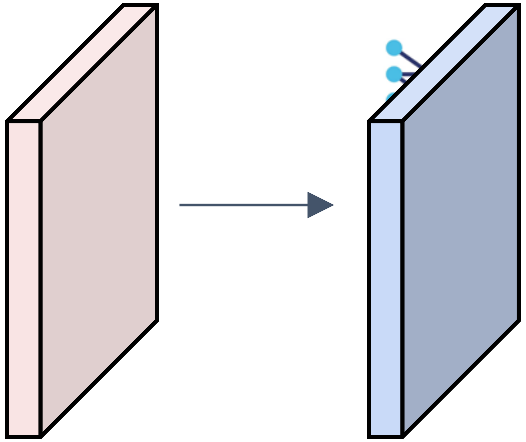
32x32x10



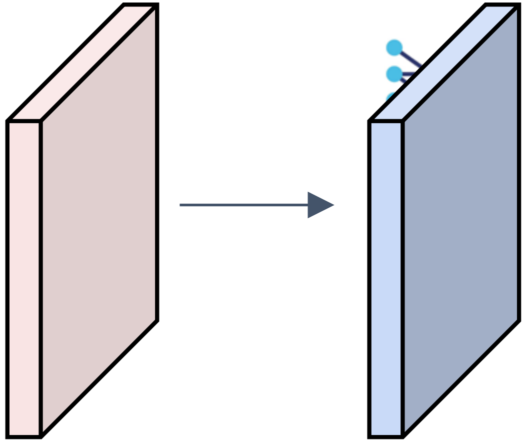
Examples time:

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

Number of parameters in this layer?



Examples:



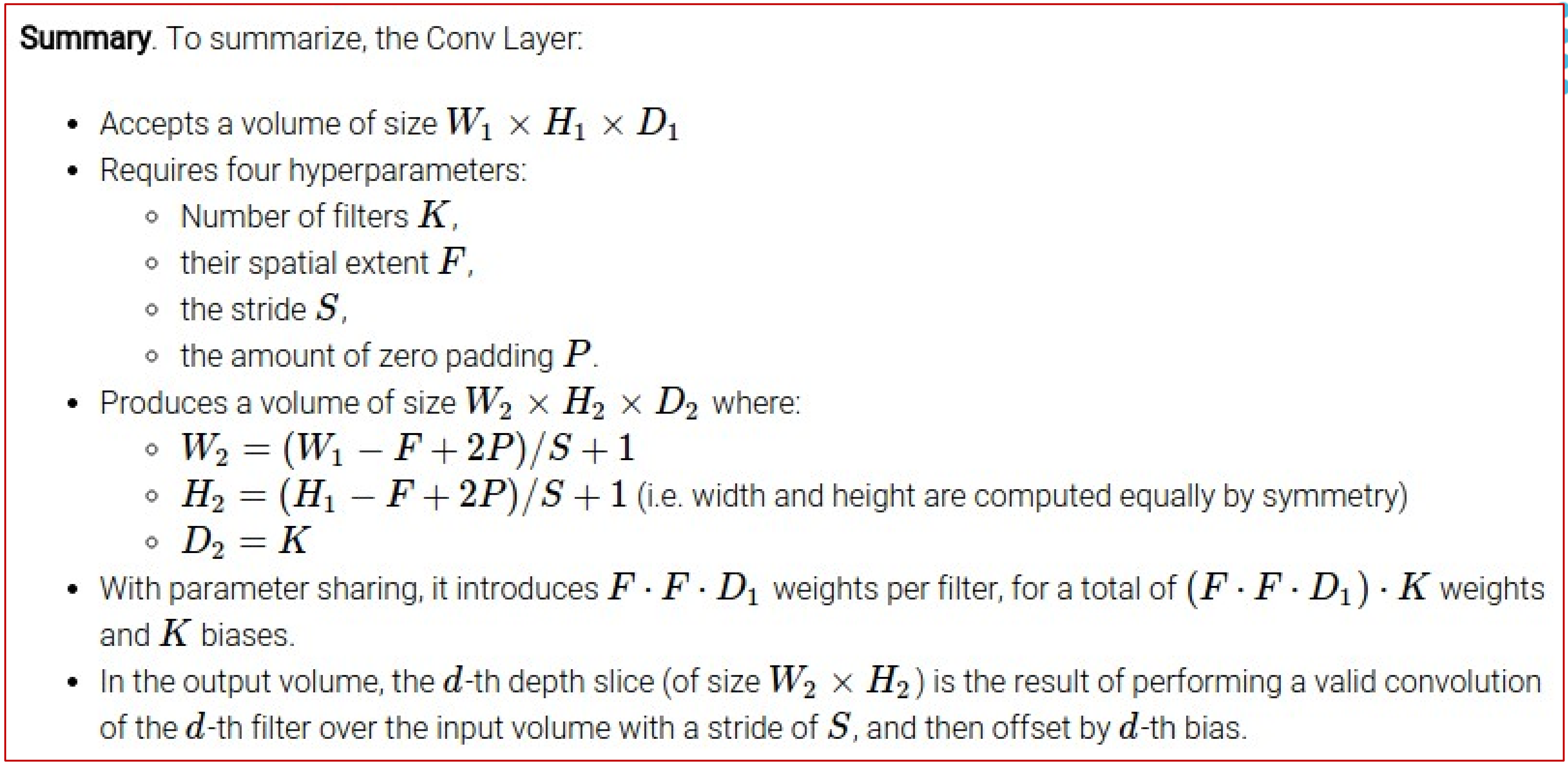
Input volume: **32x32x3**

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)

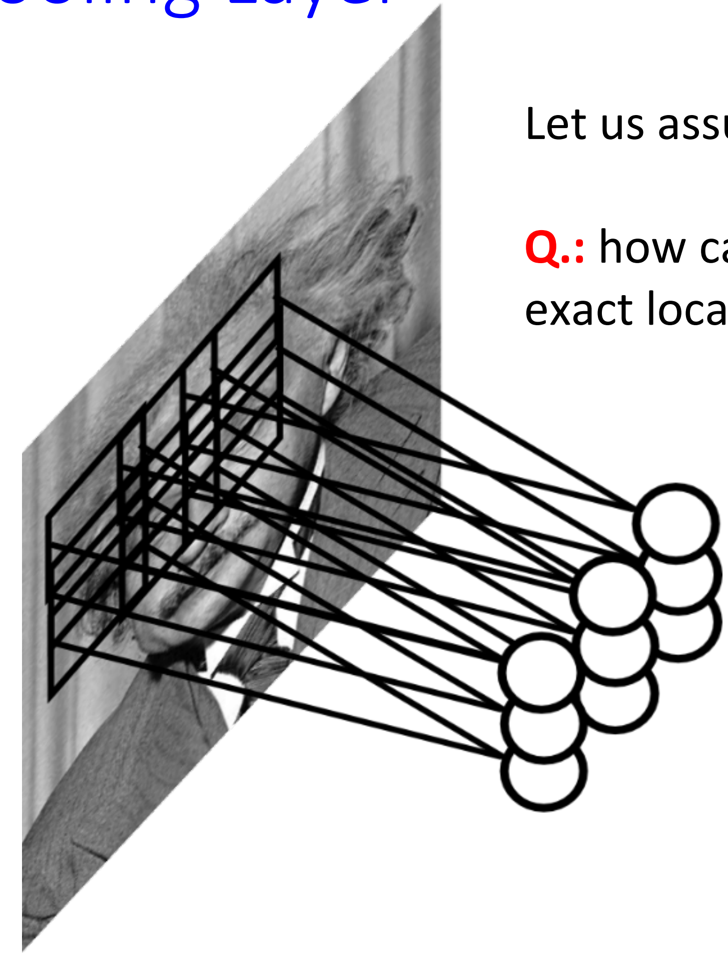
=> 76*10 = **760**



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.

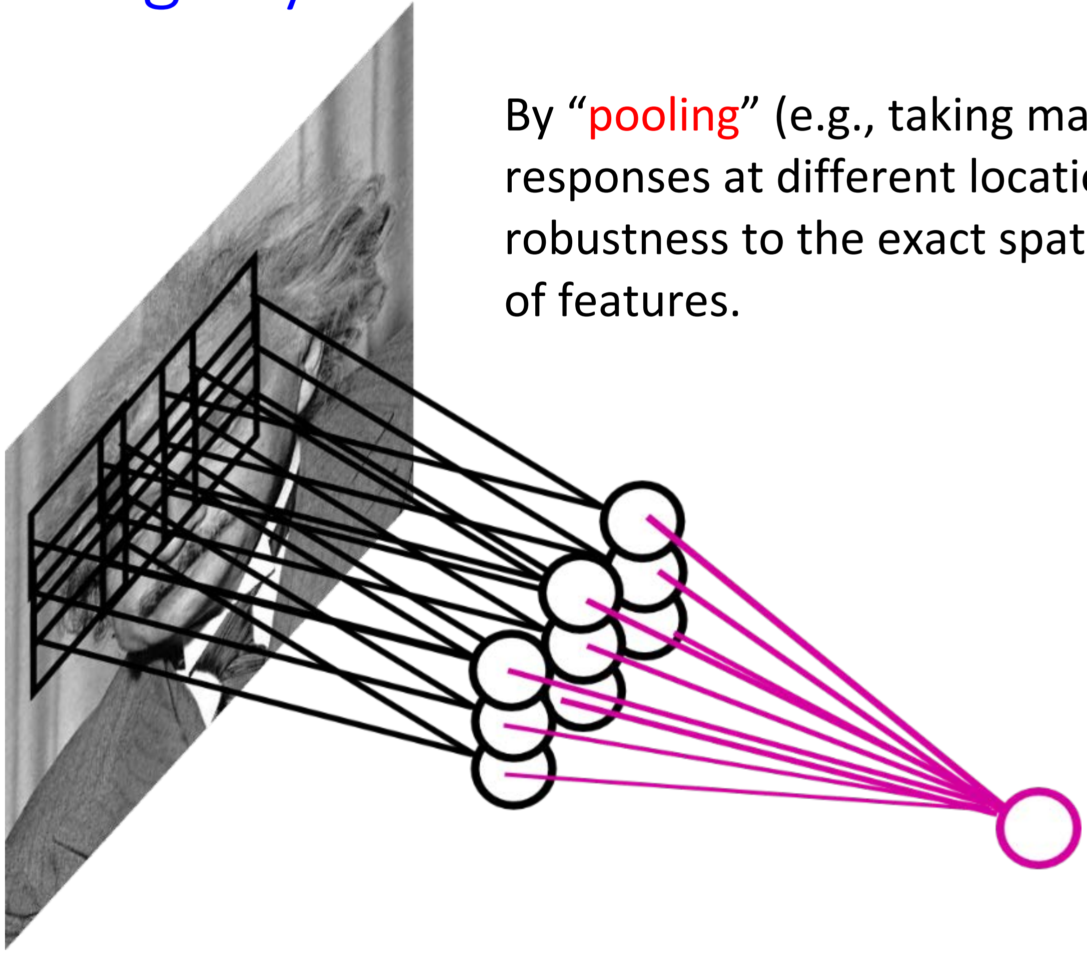
Pooling Layer



Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

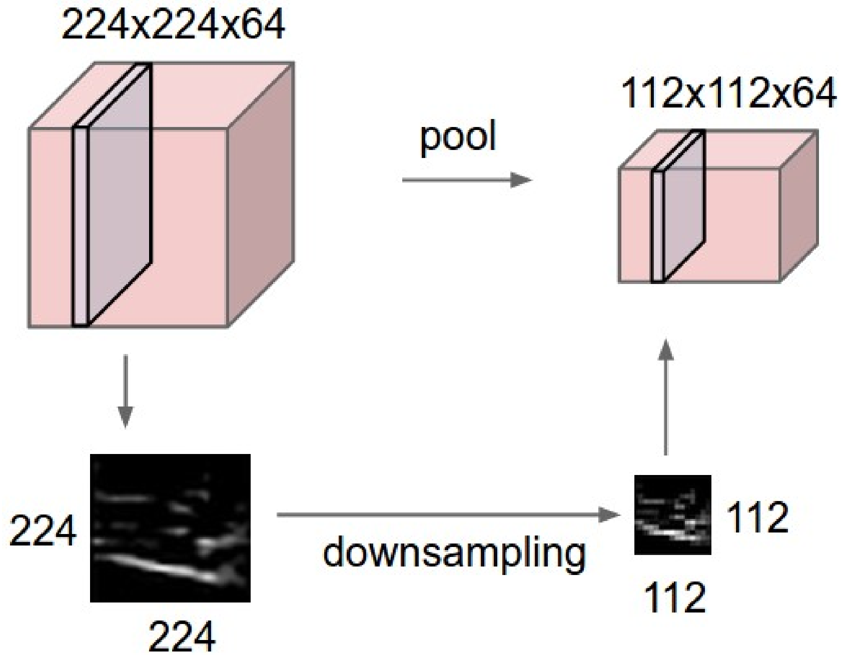
Pooling Layer



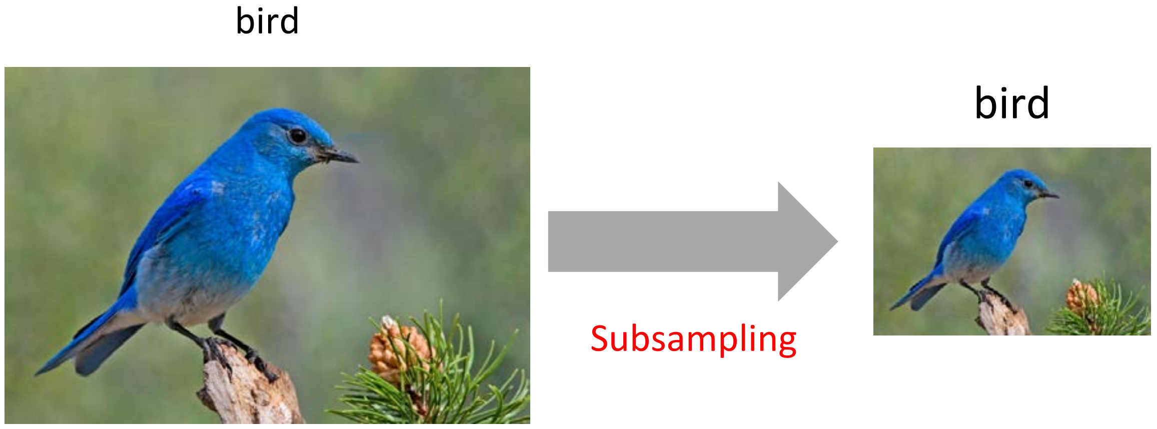
By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

Pooling layer

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



Why Pooling



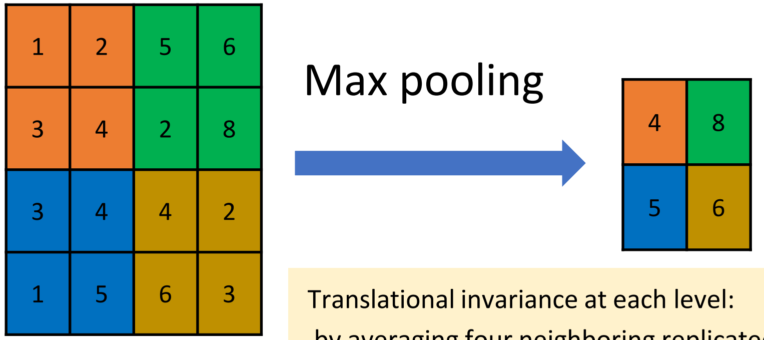
Subsampling pixels will not change the object.

We can subsample the pixels to make image smaller

Insertion of pooling layer:

- reduce the spatial size of the representation
- reduce the amount of parameters and computation in the network, and hence also control overfitting.
- The depth dimension remains unchanged.

Pool layers



Single depth slice

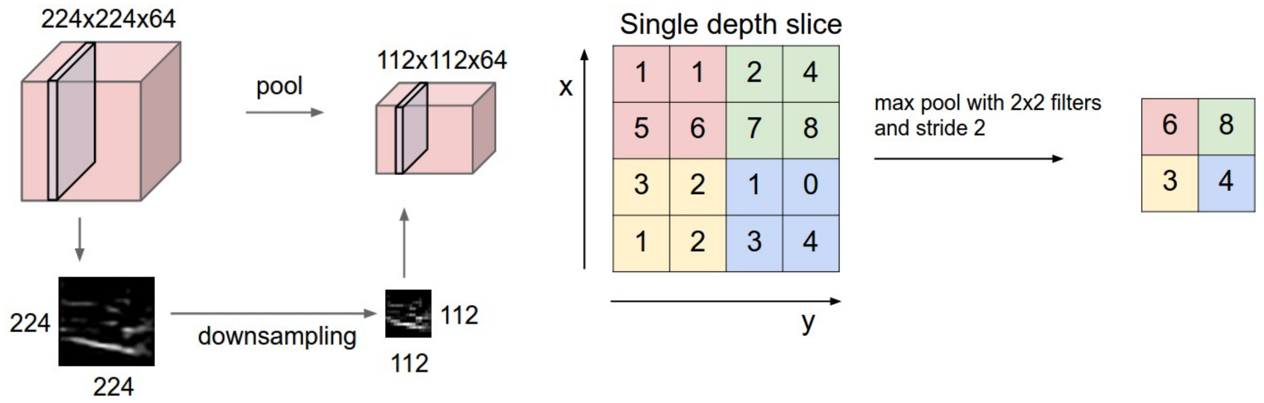
Translational invariance at each level:
by averaging four neighboring replicated detectors to give a single output to the next level.

- There are two types of the pool layers.
  - If window size = stride, this is traditional pooling.
  - If window size > stride, this is overlapping pooling.
- The larger window size and stride will be very destructive.

Problem: After several levels of pooling, we have lost information about the precise positions of things.
This makes it impossible to use the precise spatial relationships between high-level parts for recognition.

General pooling

- Other pooling functions: Average pooling, L2-norm pooling

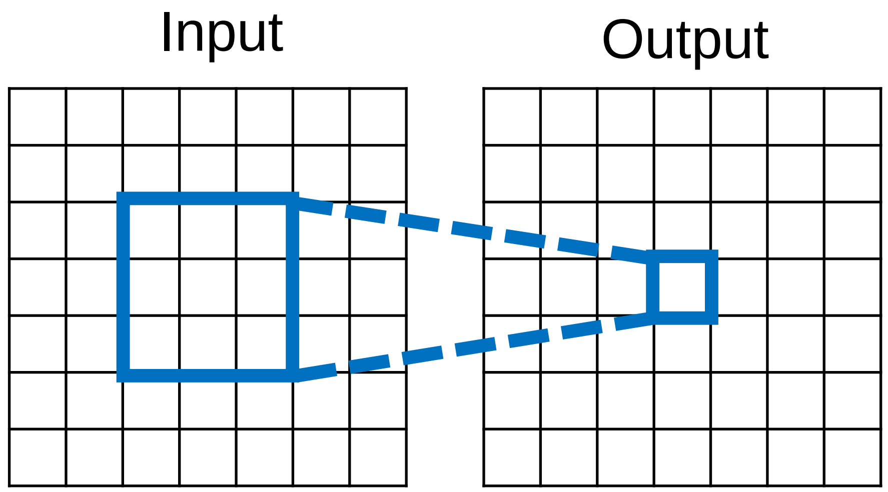


- Backpropagation. the backward pass for a max(x, y) operation routes the gradient to the input that had the highest value in the forward pass.
- Hence, during the forward pass of a pooling layer you may keep track of the index of the max activation (sometimes also called the switches) so that gradient routing is efficient during backpropagation.

Receptive field

3x3 convolutions, stride 1

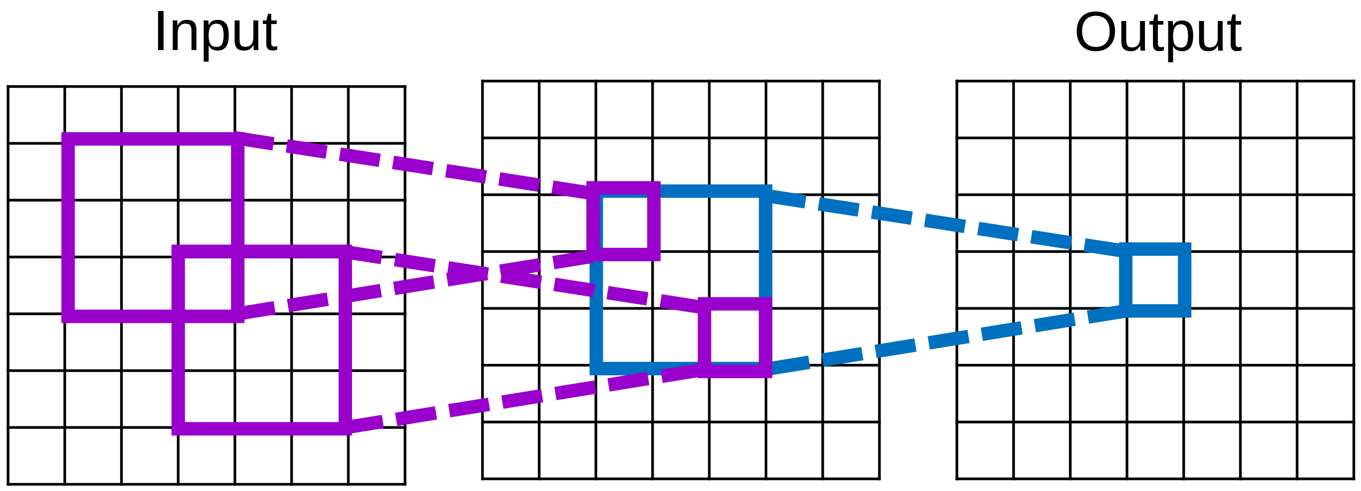
The *receptive field* of a unit is the region of the input feature map whose values contribute to the response of that unit (either in the previous layer or in the initial image)



Receptive field size: 3

Receptive field

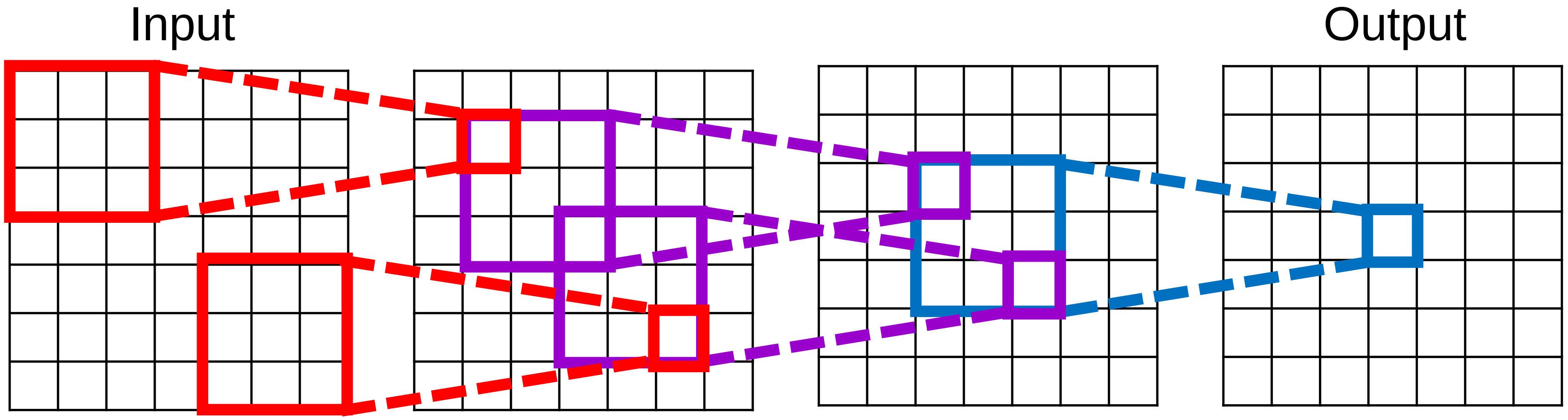
3x3 convolutions, stride 1



Receptive field size: 5

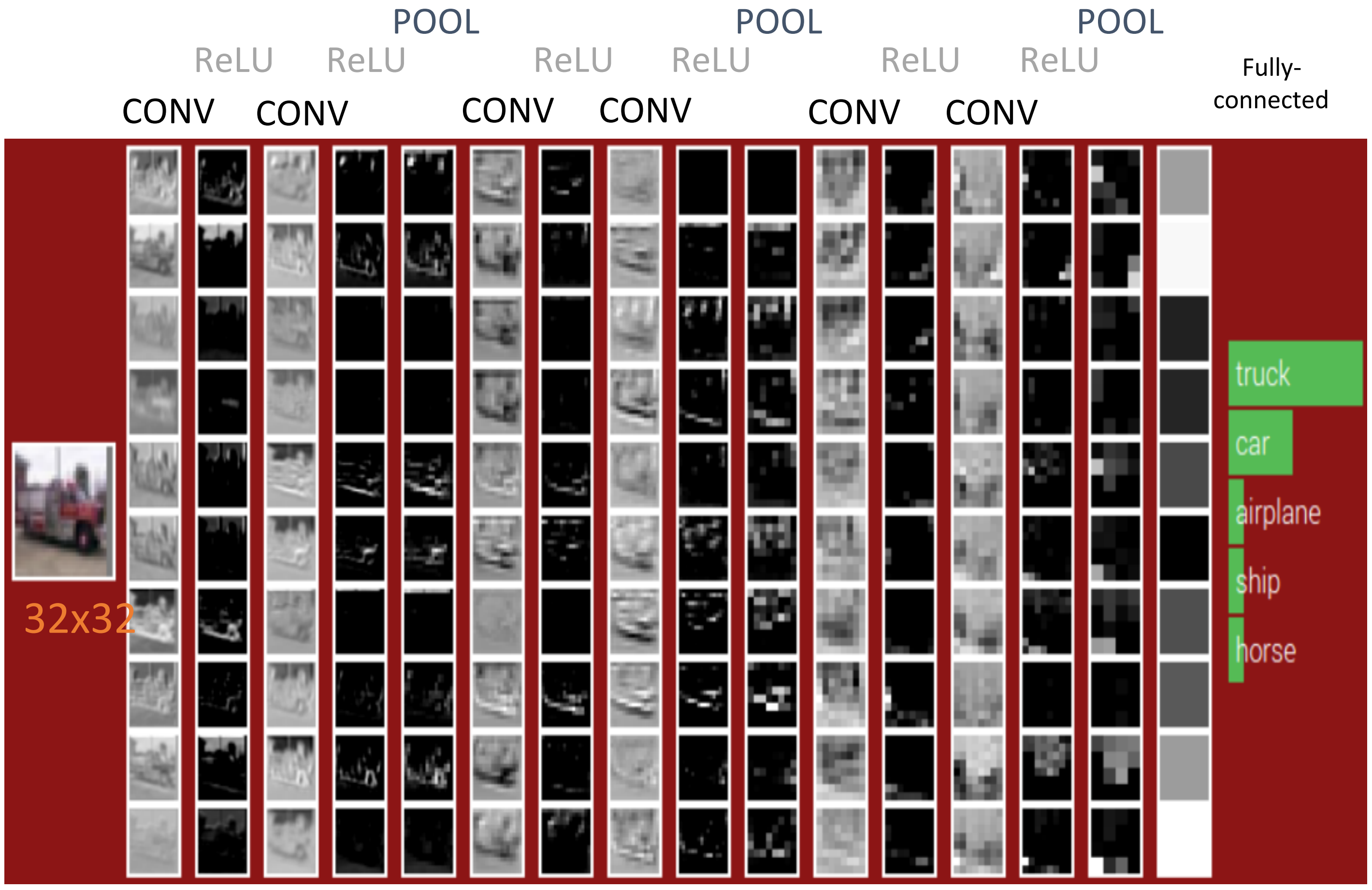
Receptive field

3x3 convolutions



Receptive field size: 7

Each successive convolution adds $F - 1$ to the receptive field size
With L layers the receptive field size is $1 + L * (F - 1)$



Weights: 280 910 910 910 910 910 1600

Size: 16x16 8x8 4x4

Getting rid of pooling

Newer networks often discard the pooling layer and have an architecture that only consists of repeated CONV layers.

- To reduce the size of the representation they suggest using larger stride in CONV layer once in a while.
- Argument:
  - The purpose of pooling layers is to perform dimensionality reduction to widen subsequent convolutional layers' receptive fields.
  - The same effect can be achieved by using a convolutional layer: using a stride of 2 also reduces the dimensionality of the output and widens the receptive field of higher layers.
- The resulting operation differs from a max-pooling layer in that
  - it cannot perform a true max operation
  - it allows pooling across input channels.

Smaller kernels

- Hand-tuning layer kernel sizes to achieve optimal receptive fields (say, 5×5 or 7×7) can be replaced by simply stacking homogenous 3×3 layers.
- The same effect of widening the receptive field is then achieved by layer composition rather than increasing the kernel size
  - three stacked 3×3 have a 7×7 receptive field.
  - At the same time, the number of parameters is reduced:
  - a 7×7 layer has 81% more parameters than three stacked 3×3 layers.

Convnets Summary

- Convolution layers represent local, shift-invariant operations on data blocks.
- Layer activations commonly organized into 4D blocks, NCHW or NHWC.
- Convolutional filters are also usually 4D arrays, e.g. $C_{out} C_{in} F_H F_W$
- Convnets typically follow conv layers with ReLU non-linearities.
- Spatial resolution decreases through the network, while number of channels increases.
- Pooling layers or strided convolutions reduce the spatial resolution in steps.