上海科技大学
ShanghaiTech University

Lecture 14: CNNs I – Architectures

Yujiao Shi
SIST, ShanghaiTech
Spring, 2025

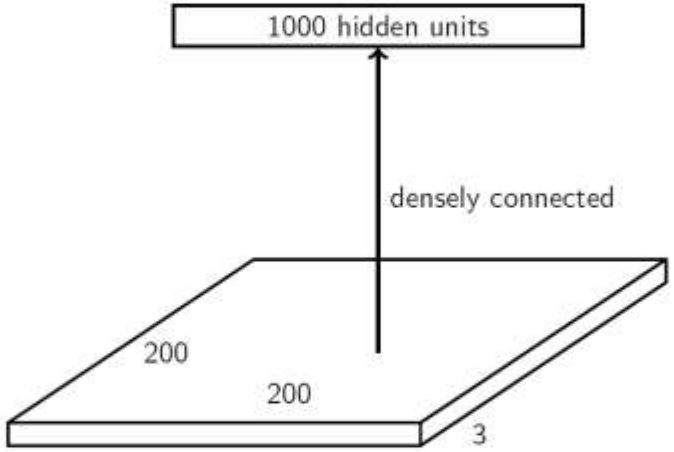
Outline

- Why Convolutional Neural Network (CNN)?
  - Motivation and overview
- What is the CNN?
  - Convolution layers & model complexity
  - Closer look at activation functions
  - Pooling layers & model complexity
  - Math properties
- Examples of CNNs

Acknowledgement: Roger Grosse@UofT & Feifei Li's cs231n notes

Motivation

- Visual recognition
  - Suppose we aim to train a network that takes a 200x200 RGB image as input



  - What is the problem with have full connections in the first layer?
    - Too many parameters! 200x200x3x1000 = 120 million
    - What happens if the object in the image shifts a little?

Our goal

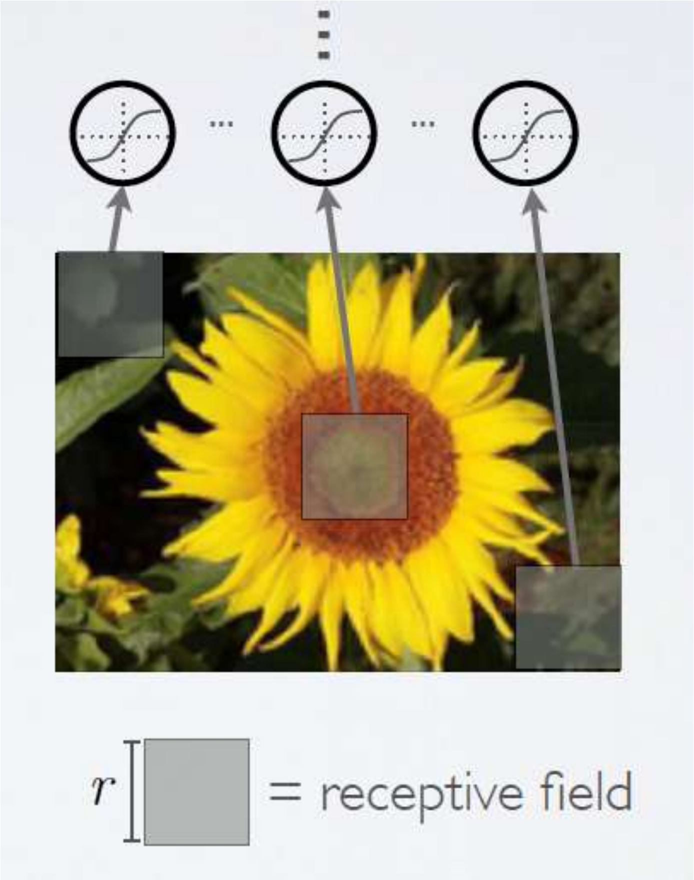
上海科技大学
ShanghaiTech University

- Visual Recognition: Design a neural network that
  - Much deal with very high-dimensional inputs
  - Can exploit the 2D topology of pixels in images
  - Can build in invariance/equivariance to certain variations we can expect
    - Translation, small deformations, illumination, etc.

- Convolution networks leverage these ideas
  - Local connectivity
  - Parameter sharing
  - Pooling/subsampling hidden units

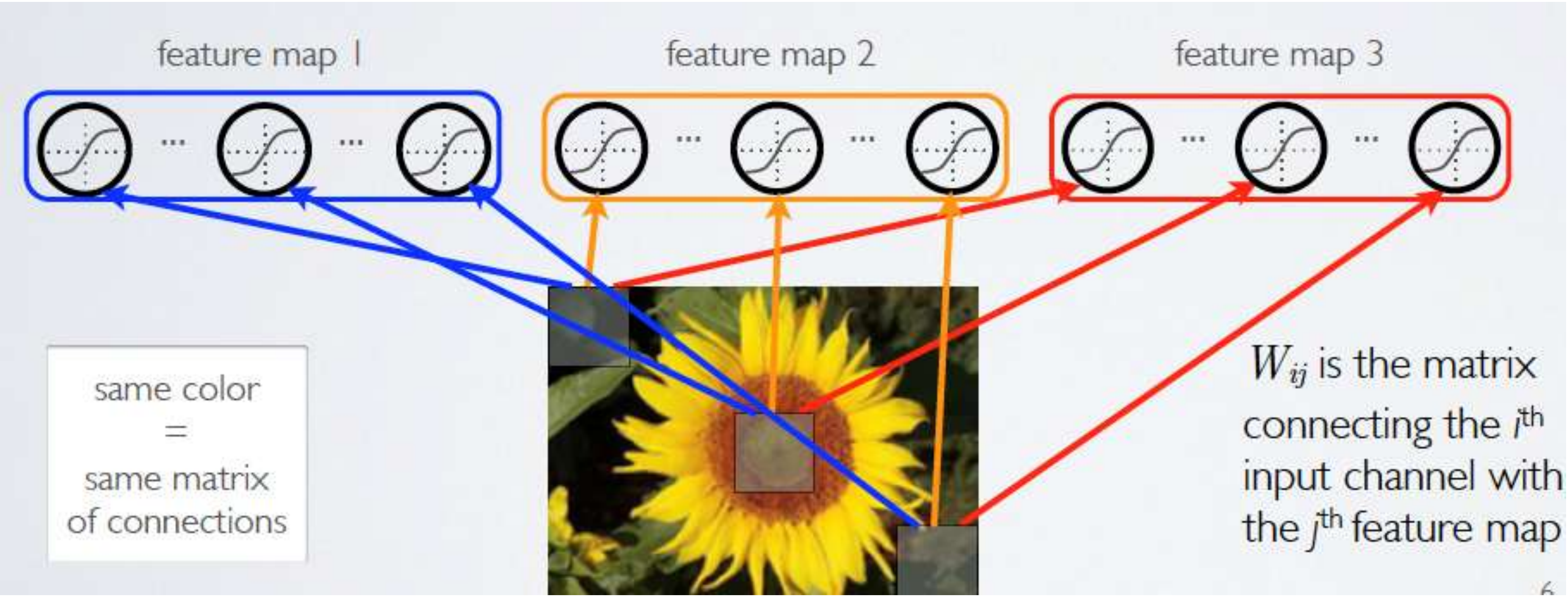
Overview of CNNs

- First idea: Use a local connectivity of hidden units
  - Each hidden unit is connected only to a subregion (patch) of the input image
  - Usually it is connected to all channels
  - Each neuron has a local receptive field



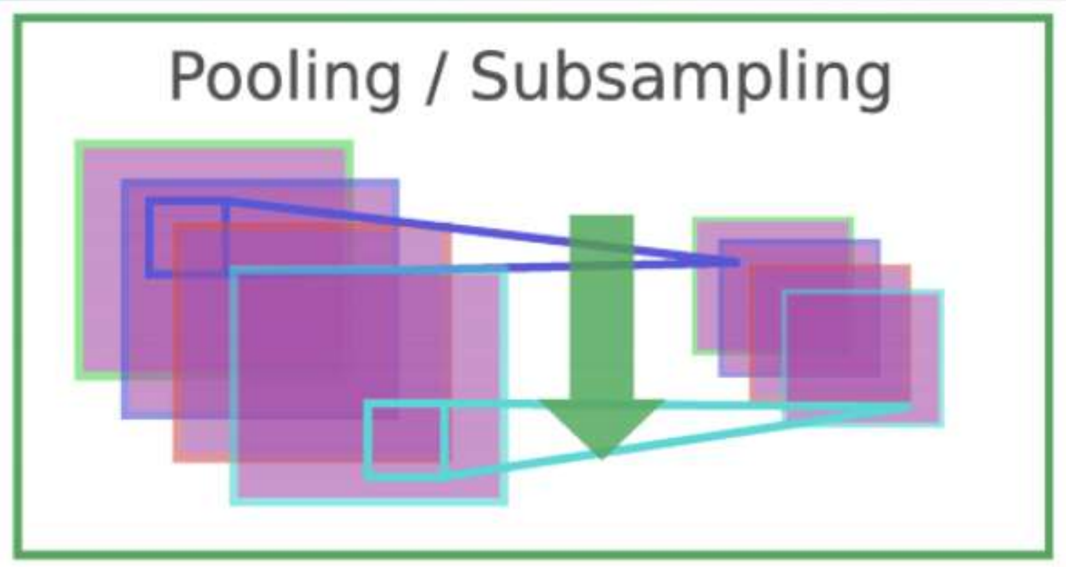
Overview of CNNs

- Second idea: share weights across certain units
  - Units organized into the same "feature map" share weight parameters
  - Hidden units within a feature map cover different positions in the image



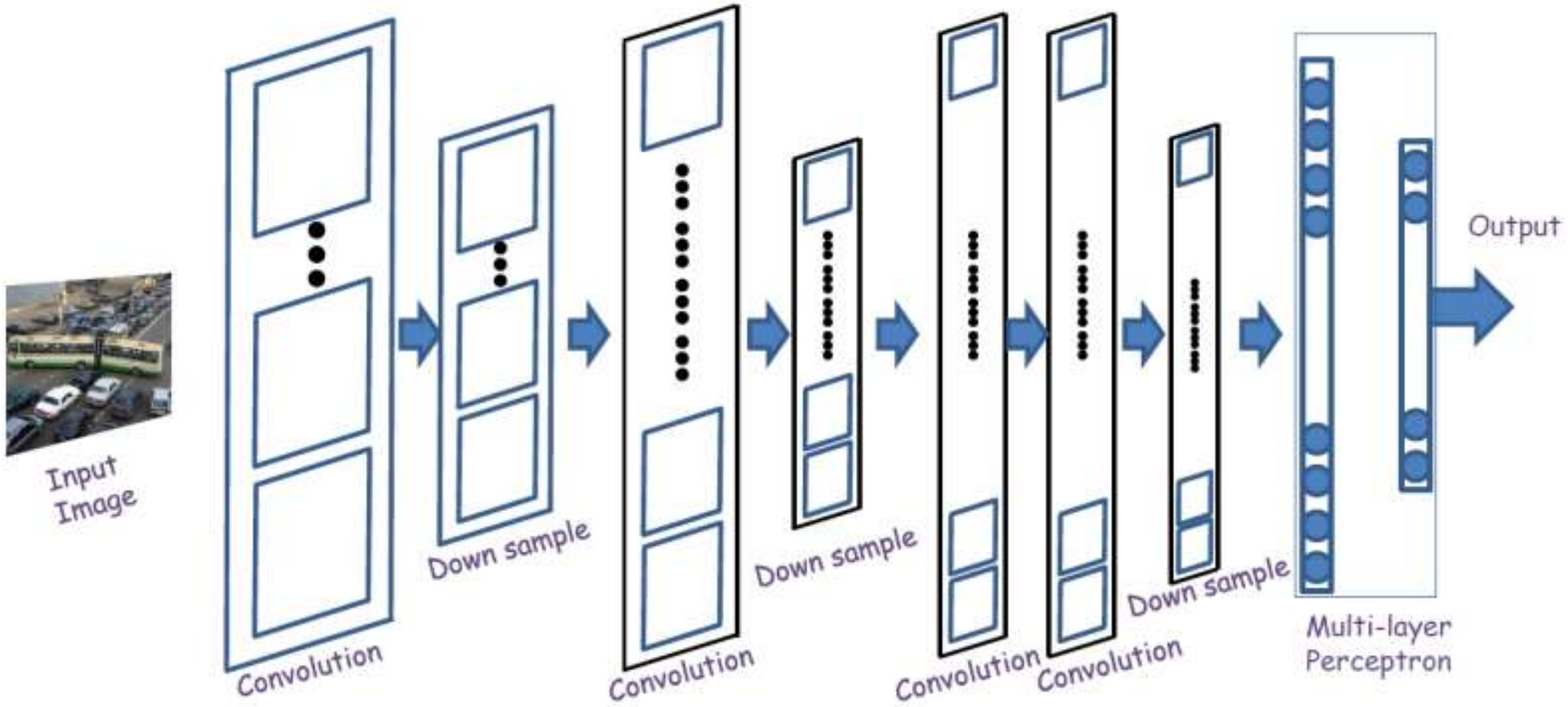
Overview of CNNs

- Third idea: pool hidden units in the same neighborhood
  - Averaging or Discarding location information in a small region
  - Robust toward small deformations in object shapes by ignoring details.



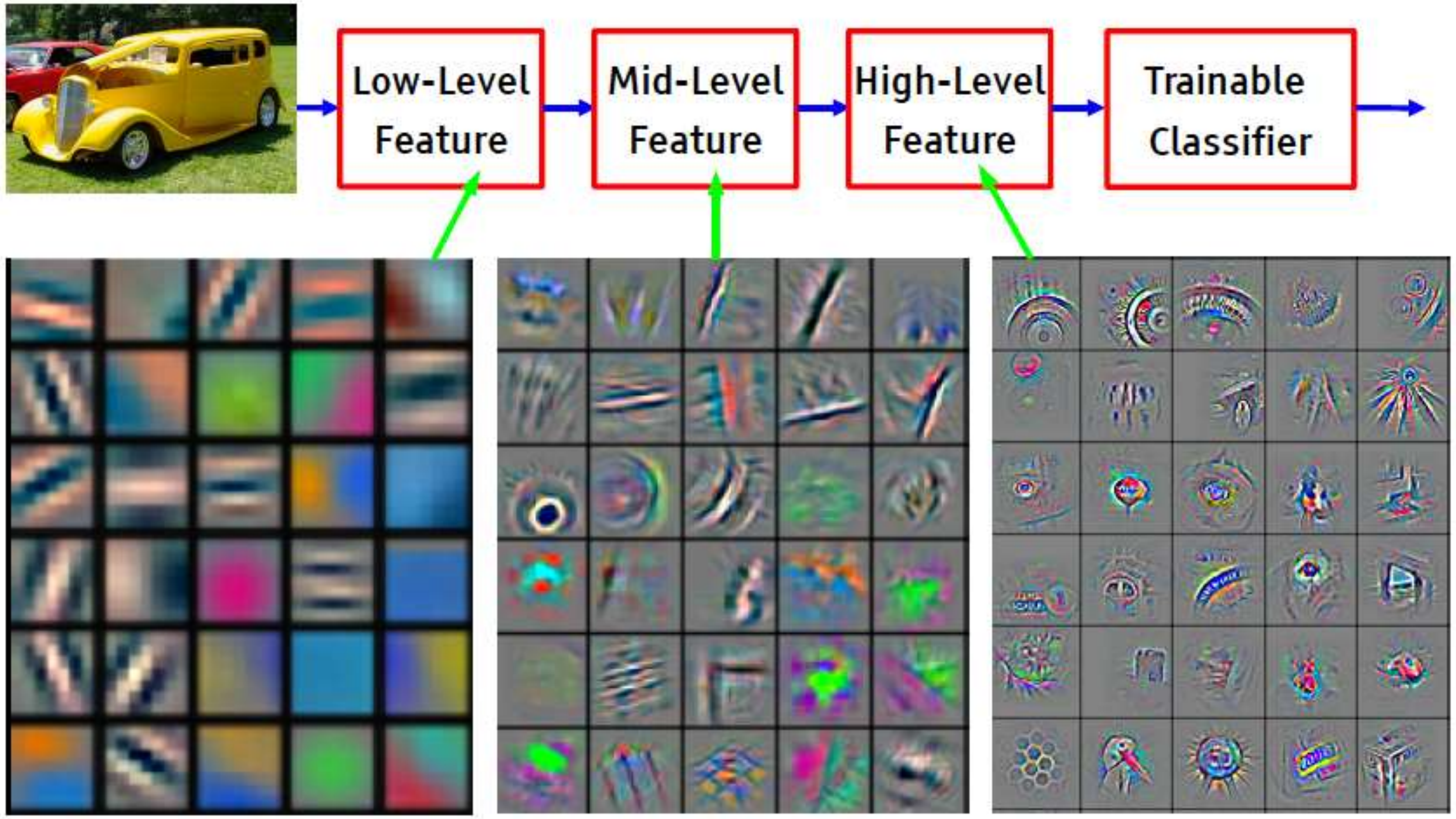
Overview of CNNs

- Fourth idea: Interleaving feature extraction and pooling operations
  - Extracting abstract, compositional features for representing semantic object classes



Overview of CNNs

- Artificial visual pathway: from images to semantic concepts (Representation learning)



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

Outline

- Why Convolutional Neural Network (CNN)?
  - Motivation and overview
- What is the CNN?
  - Convolution layers & model complexity
  - Closer look at activation functions
  - Pooling layers & model complexity
  - Math properties
- Examples of CNNs

Acknowledgement: Roger Grosse@UofT & Feifei Li's cs231n notes

2D Convolution

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 2 & 1 & 0 \\
0 & 2 & 4 & 4 & 2 & 0 \\
0 & 1 & 3 & 3 & 1 & 0 \\
0 & 1 & 2 & 3 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
$$

2D Convolution

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 1 & 2 & 2 & 1 & 0 \\
\hline
0 & 2 & 4 & 4 & 2 & 0 \\
\hline
0 & 1 & 3 & 3 & 1 & 0 \\
\hline
0 & 1 & 2 & 3 & 1 & 0 \\
\hline
0 & 0 & 1 & 1 & 0 & 0 \\
\hline
\end{array}
*
\begin{array}{|c|c|c|}
\hline
0 & 1 & 0 \\
\hline
1 & -4 & 1 \\
\hline
0 & 1 & 0 \\
\hline
\end{array}
=
\begin{array}{|c|c|c|c|}
\hline
0 & & & \\
\hline
 & & & \\
\hline
 & & & \\
\hline
 & & & \\
\hline
\end{array}
$$

$$
(0*0)+(0*1)+(0*0)+(0*1)+(1*-4) \\
+(2*1)+(0*0)+(2*1)+(4*0)=0
$$

2D Convolution

上海科技大学 ShanghaiTech University

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

$$\begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 2 & 2 & 1 & 0 \\ \hline 0 & 2 & 4 & 4 & 2 & 0 \\ \hline 0 & 1 & 3 & 3 & 1 & 0 \\ \hline 0 & 1 & 2 & 3 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & -1 & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

$$(0*0)+(0*1)+(0*0)+(1*1)+(2*-4)$$
$$+(2*1)+(2*0)+(4*1)+(4*0)=-1$$

2D Convolution

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 1 & 2 & 2 & 1 & 0 \\ \hline
0 & 2 & 4 & 4 & 2 & 0 \\ \hline
0 & 1 & 3 & 3 & 1 & 0 \\ \hline
0 & 1 & 2 & 3 & 1 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 0 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
0 & 1 & 0 \\ \hline
1 & -4 & 1 \\ \hline
0 & 1 & 0 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|}
\hline
0 & -1 & -1 & 0 \\ \hline
-2 & -5 & -5 & -2 \\ \hline
2 & -2 & -1 & 3 \\ \hline
-1 & 0 & -5 & 0 \\ \hline
\end{array}
$$

Convolutional Filters

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Poll Question 1:

What effect do you think the following filter will have on an image?

A. Sharpen the image

B. Blur the image

C. Shift the image left

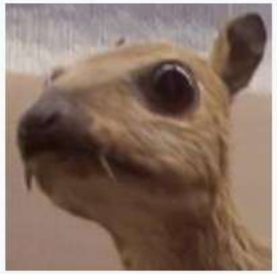
D. Rotate the image clockwise

E. Nothing (TOXIC)

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

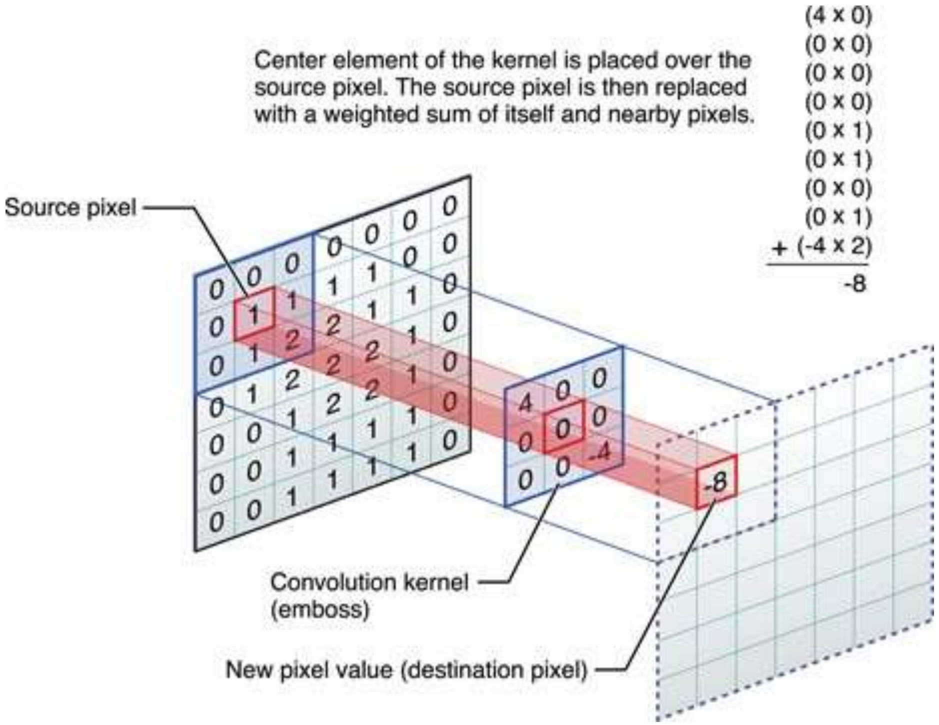
Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

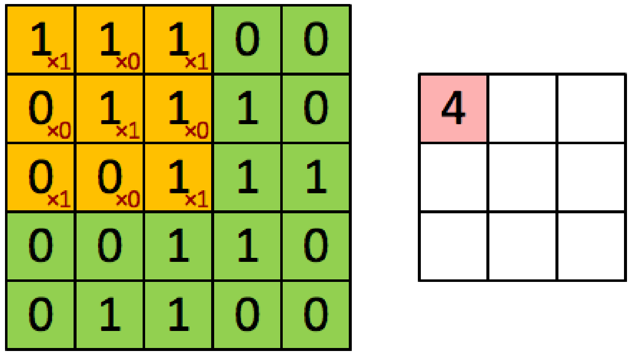
More Filters

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

2D Convolution





Image

Convolved Feature

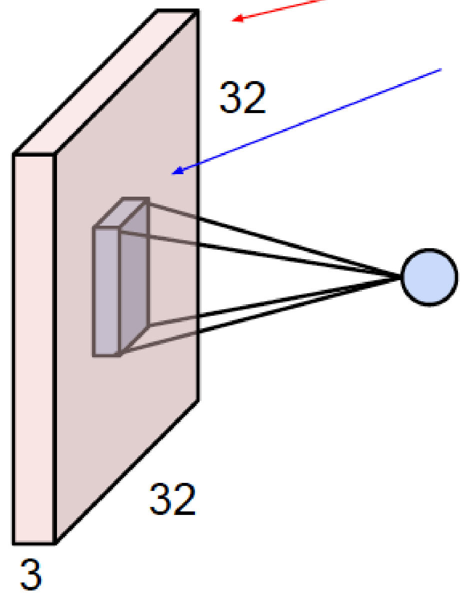
Picture Courtesy: developer.apple.com

Convolution Layers

- Formal definition

32x32x3 image

5x5x3 filter w

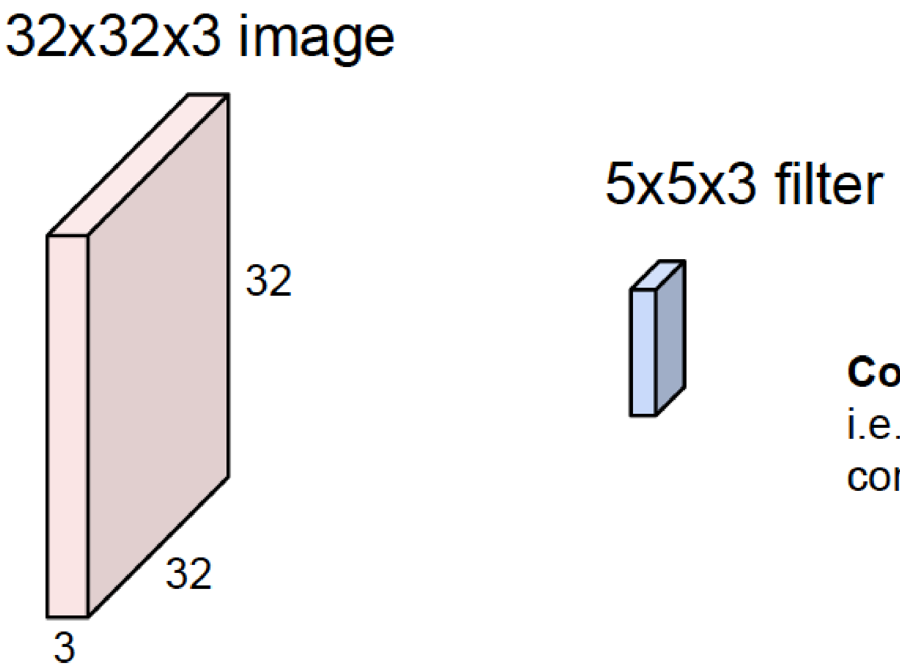


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layers

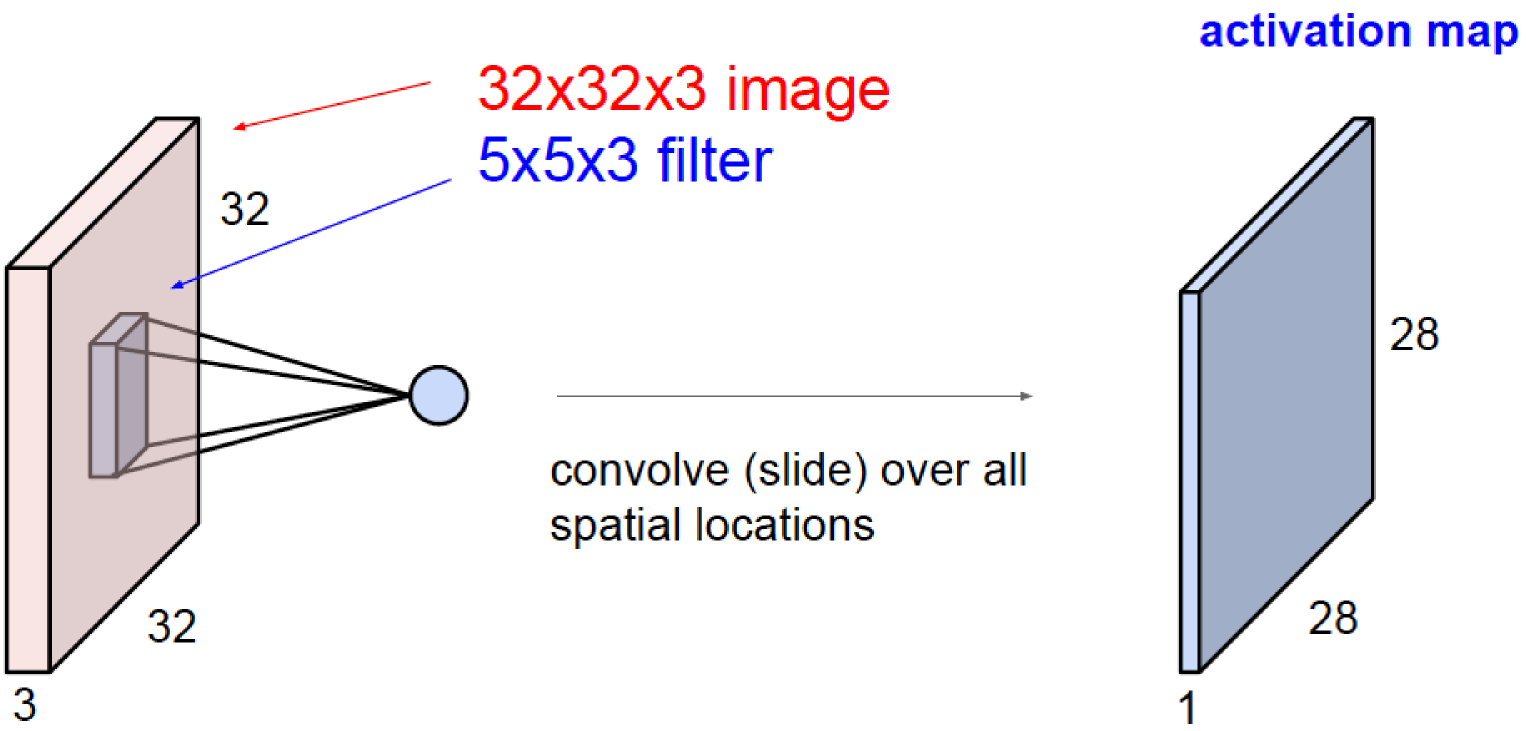
- Define a neuron corresponding to a 5x5 filter



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

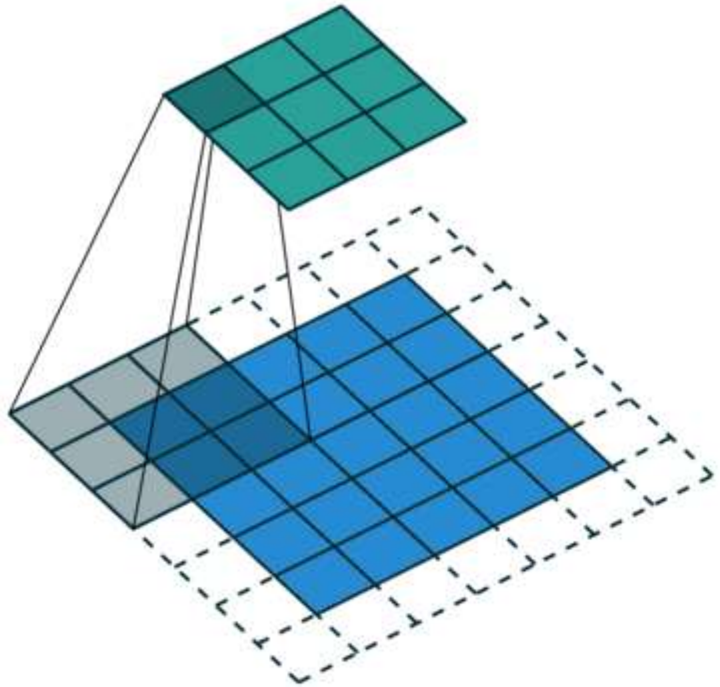
Convolution Layers

- Convolution operation
  - Parameter sharing
  - Spatial information



Convolution Layers

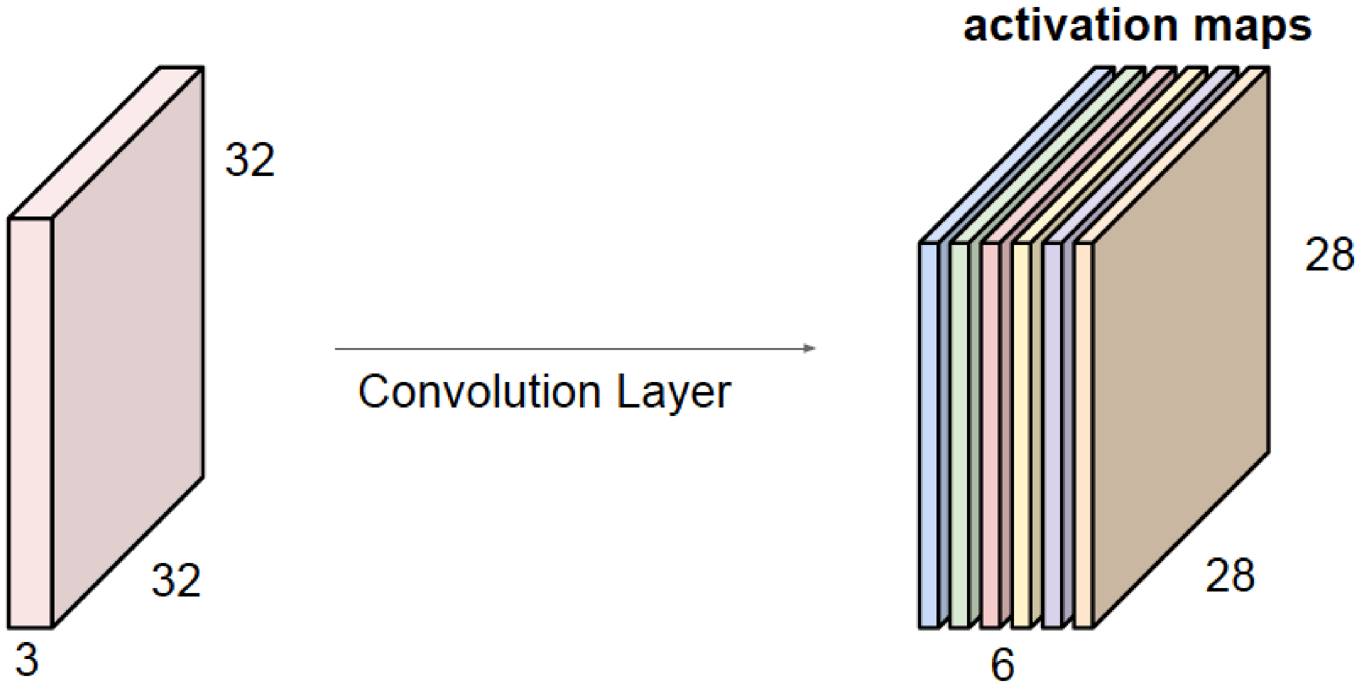
- Convolution operation
  - Parameter sharing
  - Spatial information



Convolution Layers

- Multiple kernels/filters

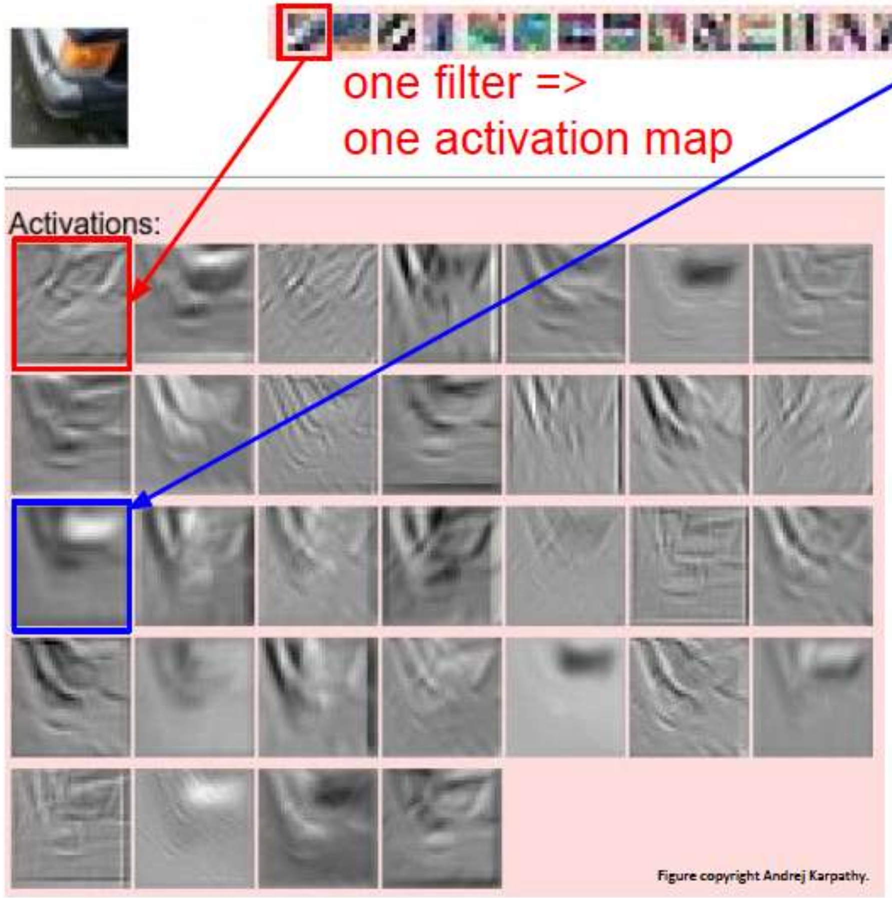
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

Convolution Layers

- Visualizing the filters and their outputs



example 5x5 filters
(32 total)

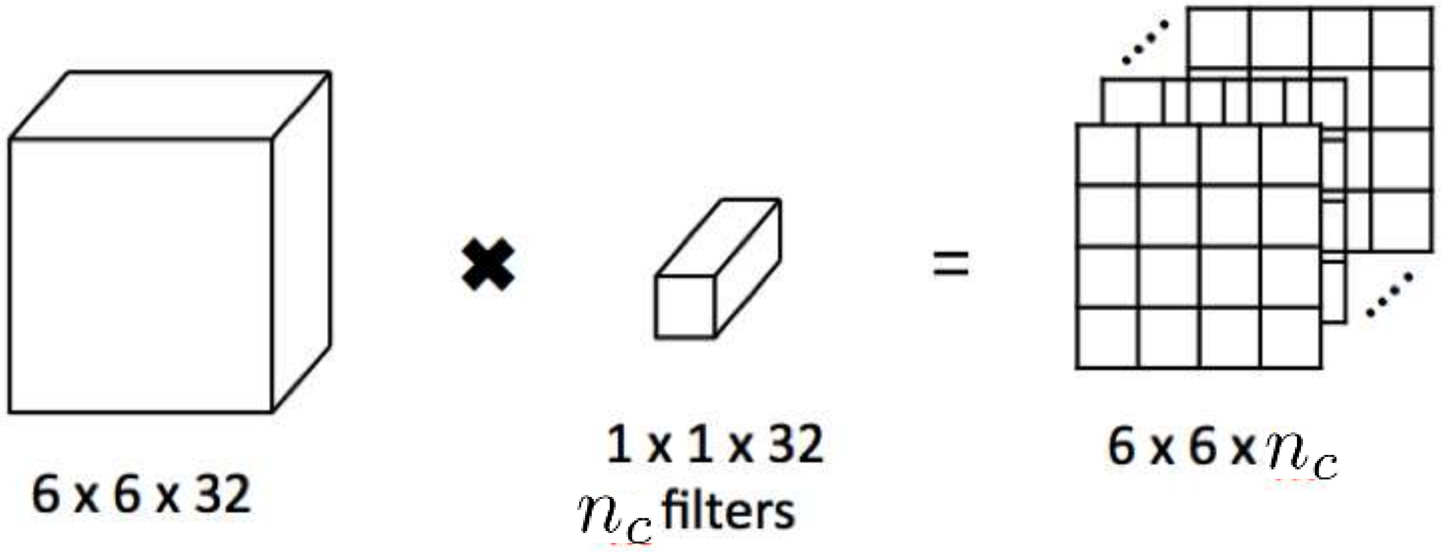
We call the layer convolutional because it is related to convolution of two signals:

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

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

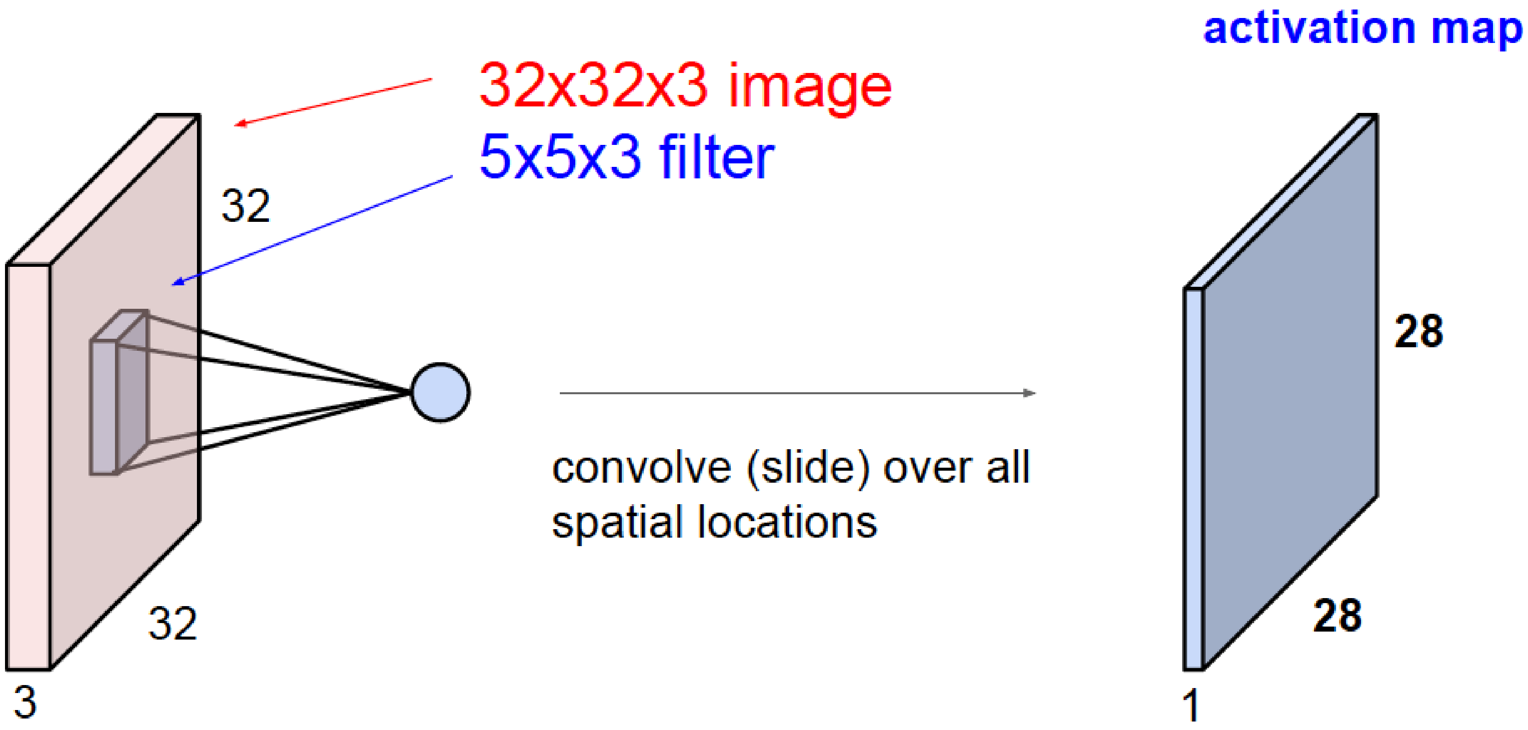
Special Convolutions

- 1x1 convolutions
  - Used in Network-in-network, GoogleNet
  - Reduce or increase dimensionality
  - Can be considered as ‘feature pooling”



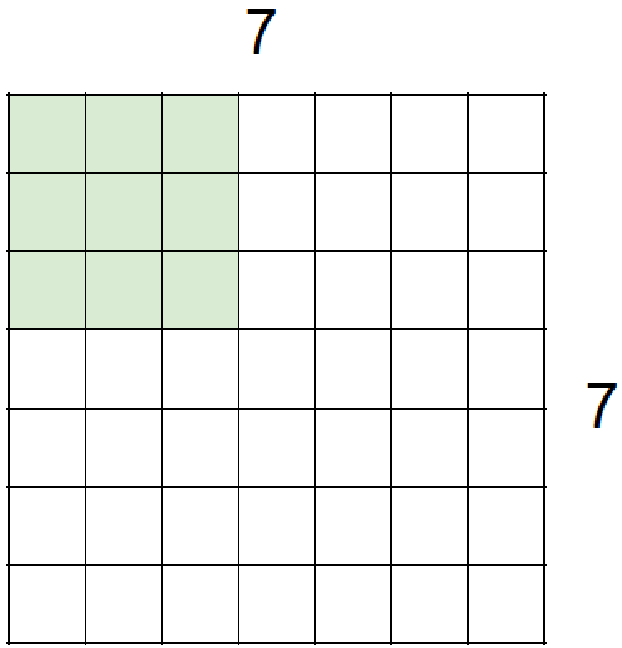
Complexity of Convolution Layers

- Sizes of activation maps and number of parameters



Complexity of Convolution Layers

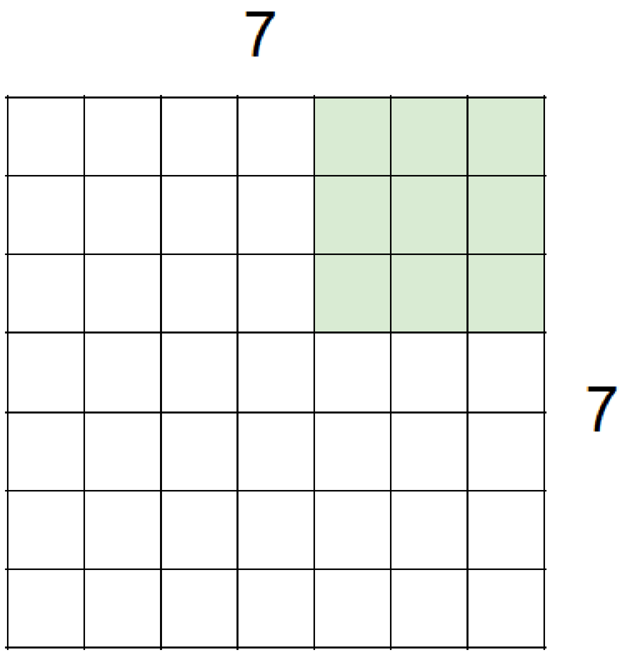
- Size of activation maps



7x7 input (spatially)
assume 3x3 filter

Complexity of Convolution Layers

- Size of activation maps

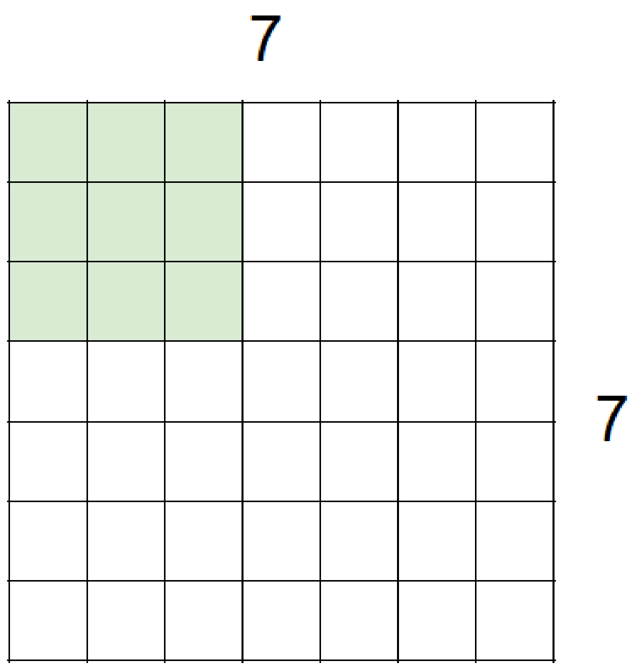


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Complexity of Convolution Layers

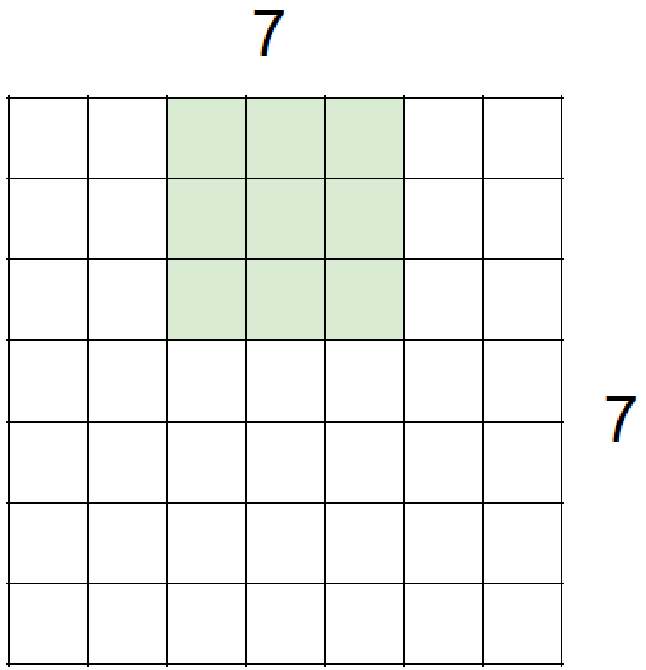
- Case: Stride > 1



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

Complexity of Convolution Layers

- Case: Stride > 1

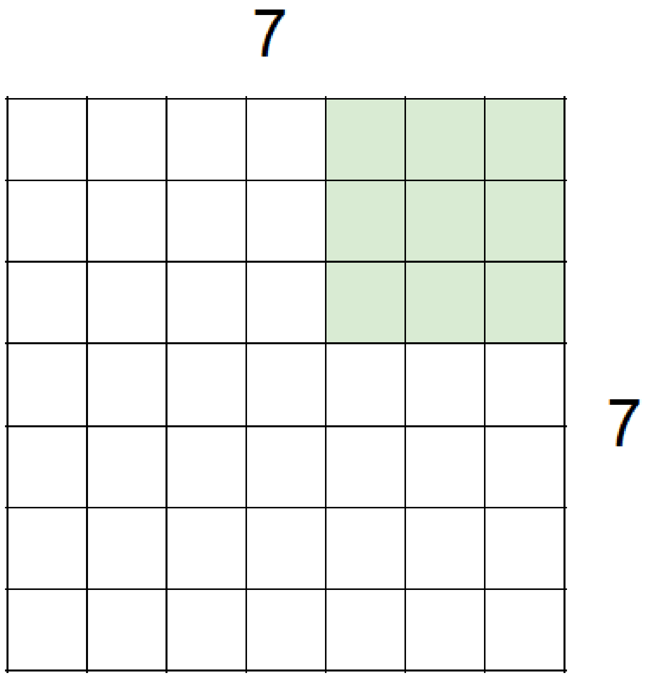


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Complexity of Convolution Layers

上海科技大学 ShanghaiTech University

- Case: Stride > 1



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

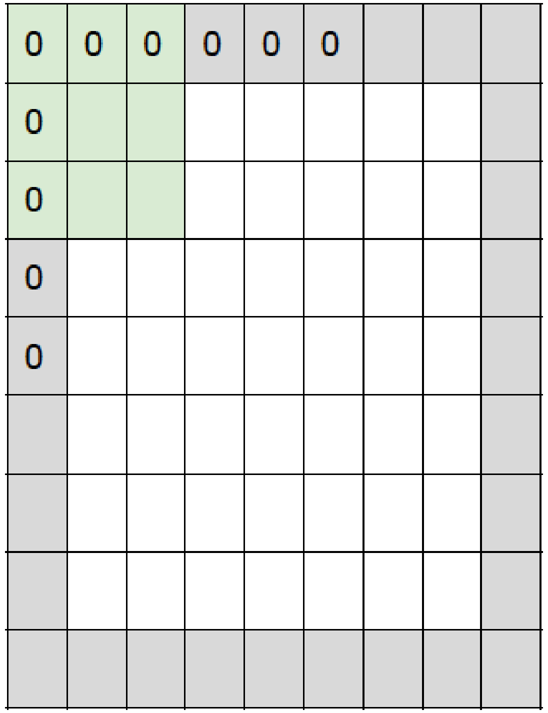
Output size:

(N-F)/stride + 1

Complexity of Convolution Layers

上海科技大学
ShanghaiTech University

- Zero padding to handle non-integer cases or control the output sizes



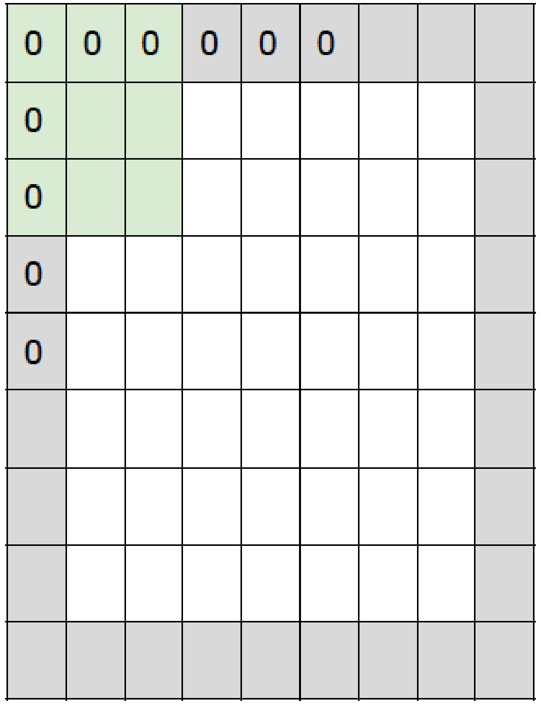
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

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

Complexity of Convolution Layers

- Zero padding to handle non-integer cases or control the output sizes

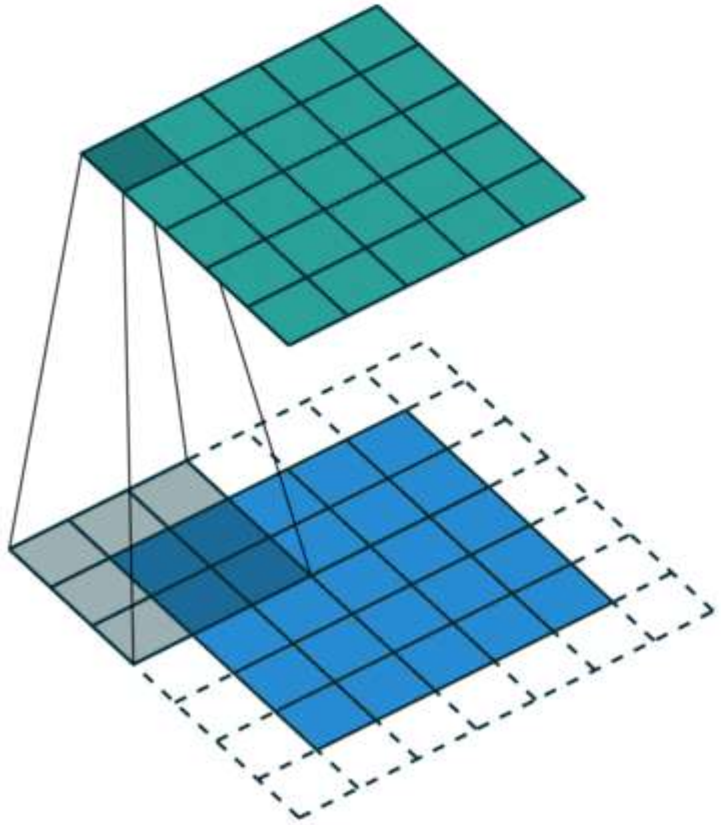


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

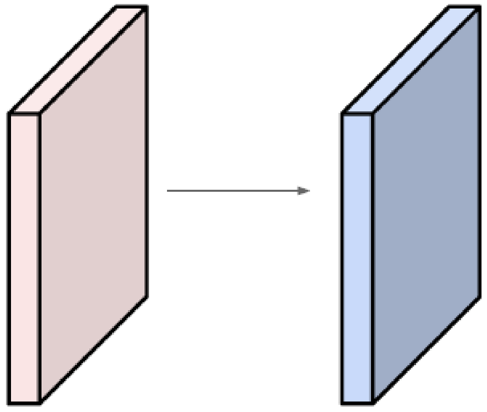
Complexity of Convolution Layers

- Zero padding to handle non-integer cases or control the output sizes



Complexity of Convolution Layers

Examples time:

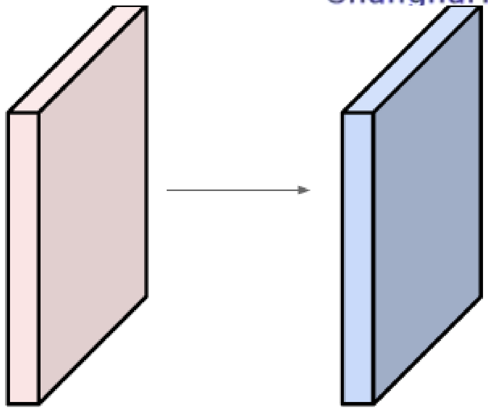


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Complexity of Convolution Layers

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**

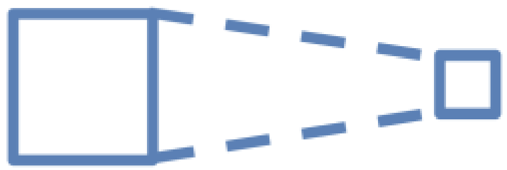
Complexity of Convolution Layers

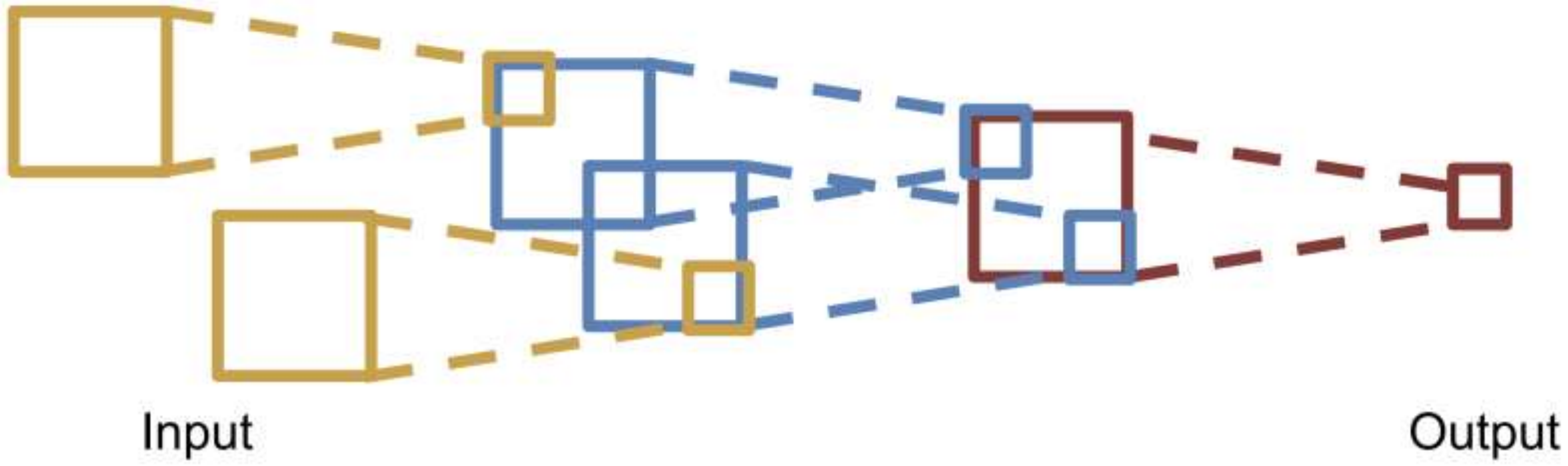
- Summary

- 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.

Receptive Fields

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





Outline

上海科技大学
ShanghaiTech University

- Why Convolutional Neural Network (CNN)?
  - Motivation and overview
- What is the CNN?
  - Convolution layers & model complexity
  - Closer look at activation functions
  - Pooling layers & model complexity
  - Math properties
- Examples of CNNs

Acknowledgement: Roger Grosse@UofT & Feifei Li's cs231n notes

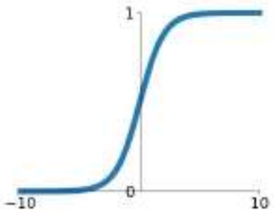
Review: Activation Function

上海科技大学 ShanghaiTech University

- Zoo of Activation functions

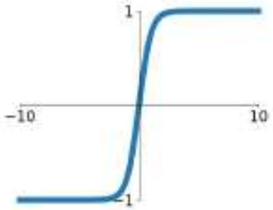
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



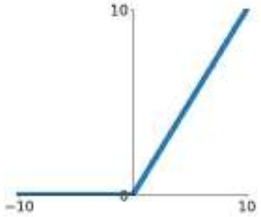
tanh

$\tanh(x)$



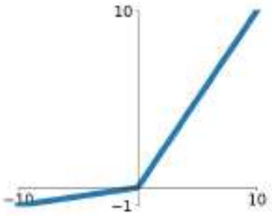
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

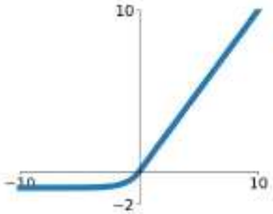


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

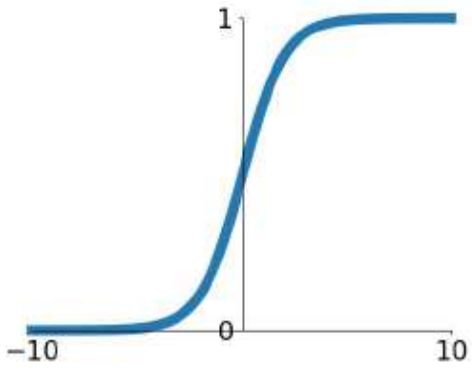
ELU

$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Sigmoid function

$$\sigma(x) = 1/(1 + e^{-x})$$



Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

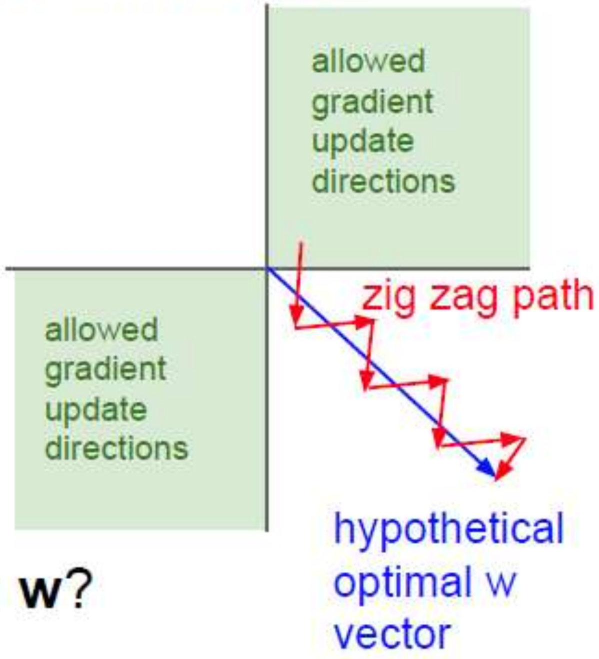
3 problems:

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

Sigmoid function

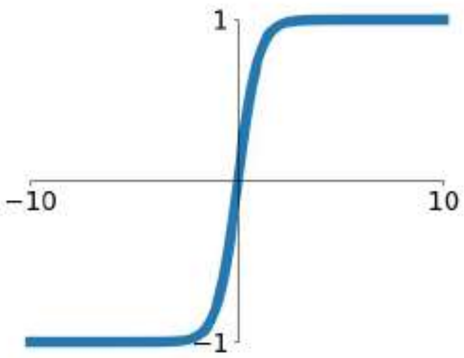
Consider what happens when the input to a neuron is always positive...

$$f\left(\sum_i w_i x_i + b\right)$$



What can we say about the gradients on **w**?
Always all positive or all negative :(
(this is also why you want zero-mean data!)

Tanh function



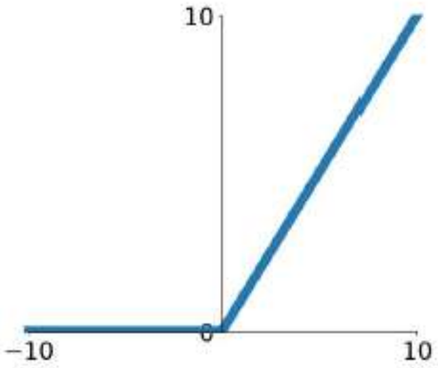
tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

[LeCun et al., 1991]

- Recurrent neural networks: LSTM, GRU

Rectified Linear Unit



ReLU
(Rectified Linear Unit)

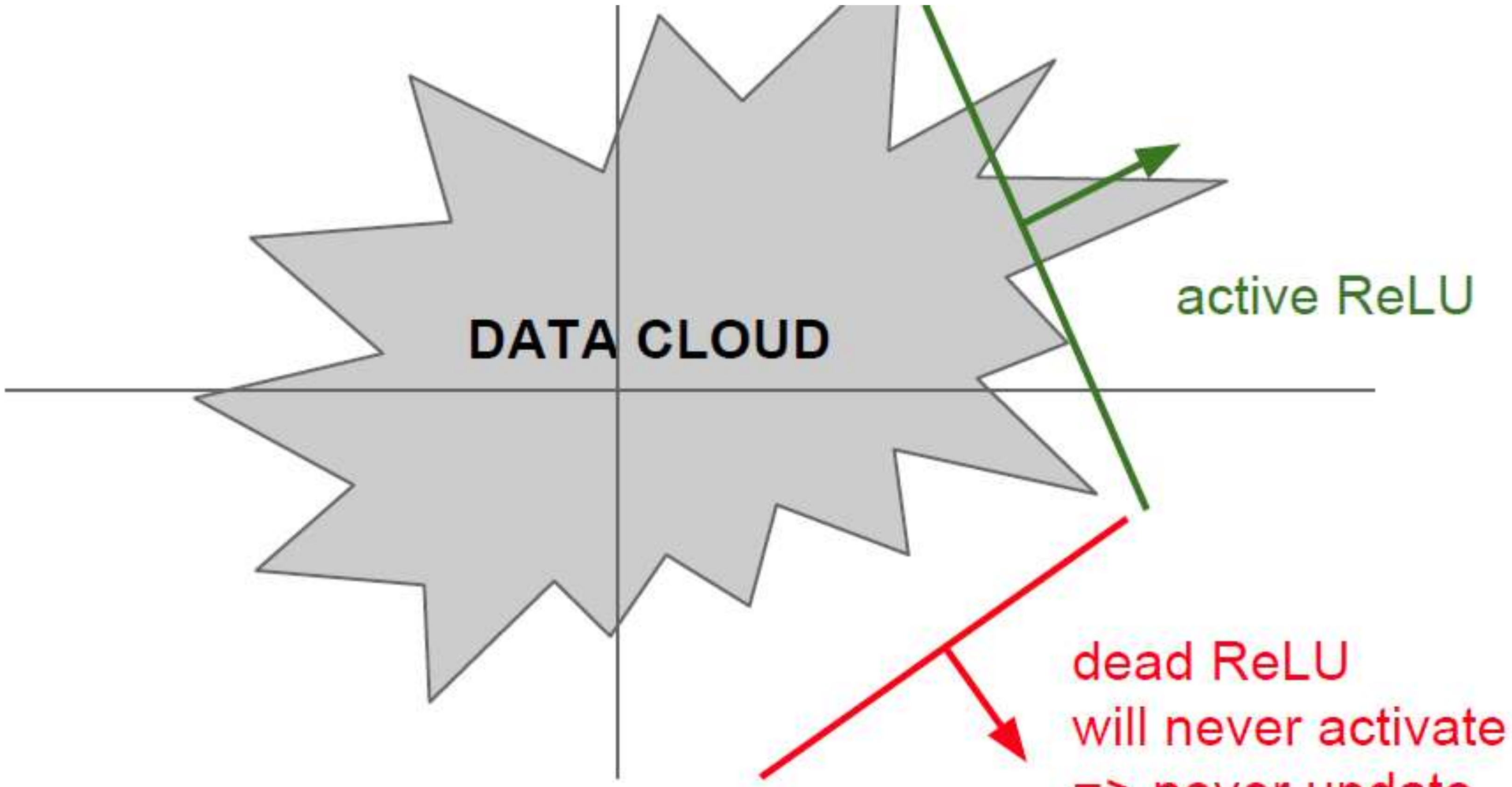
- Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

- Not zero-centered output
- An annoyance:

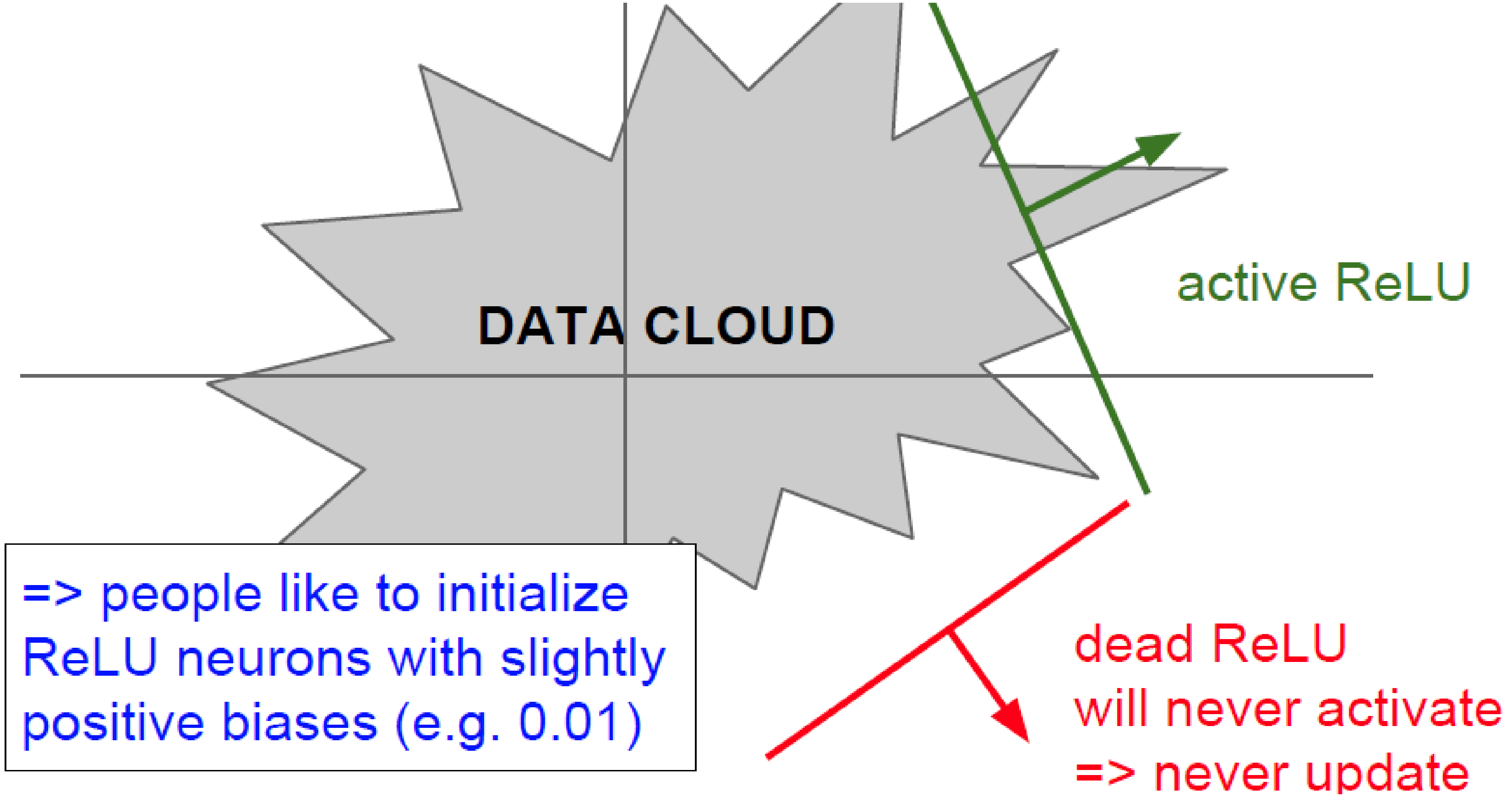
hint: what is the gradient when x < 0?

Rectified Linear Unit



Rectified Linear Unit

上海科技大学
ShanghaiTech University

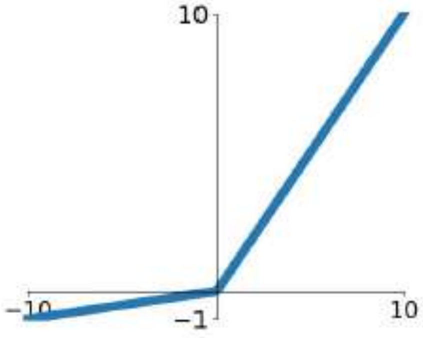


=> people like to initialize ReLU neurons with slightly positive biases (e.g. 0.01)

Leaky ReLU

Activation Functions

[Mass et al., 2013]
[He et al., 2015]



Leaky ReLU

$$f(x) = \max(0.01x, x)$$

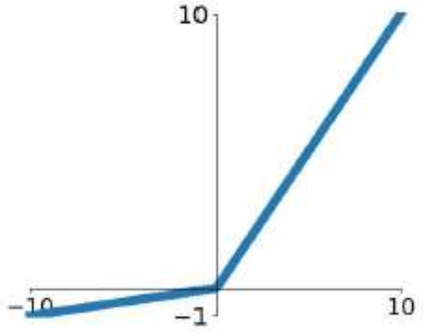
- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Leaky ReLU

上海科技大学
ShanghaiTech University

Activation Functions

[Mass et al., 2013]
[He et al., 2015]



- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Parametric Rectifier (PReLU)

$$f(x) = \max(\alpha x, x)$$

backprop into \alpha
(parameter)

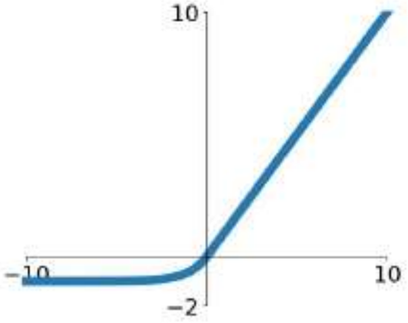
Exponential Linear Units (ELU)

上海科技大学
ShanghaiTech University

Activation Functions

[Clevert et al., 2015]

Exponential Linear Units (ELU)



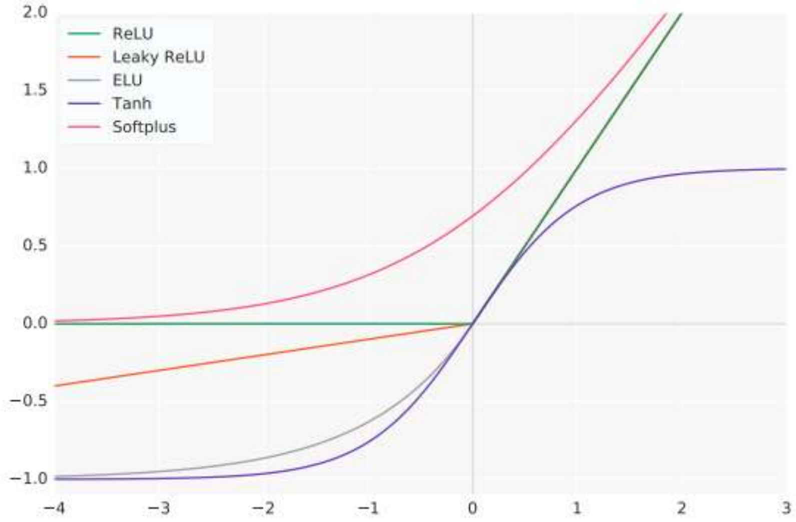
- All benefits of ReLU
- Closer to zero mean outputs
- Negative saturation regime compared with Leaky ReLU adds some robustness to noise

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$

- Computation requires exp()

Summary: Activation function

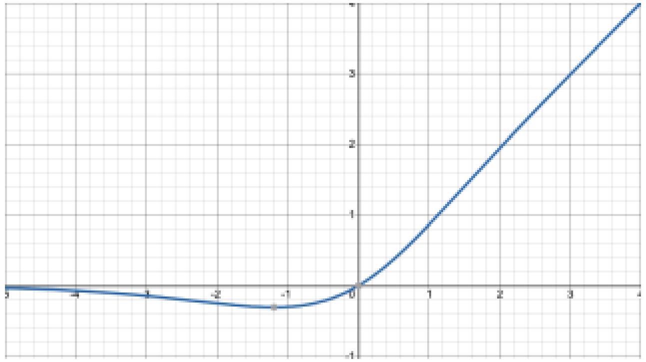
- For internal layers in CNNs
  - Use ReLU. Be careful with your learning rates
  - Try out Leaky ReLU / Maxout / ELU
  - Try out tanh but don't expect much
  - Don't use sigmoid



- For output layers
  - Task dependent
  - Related to your loss function

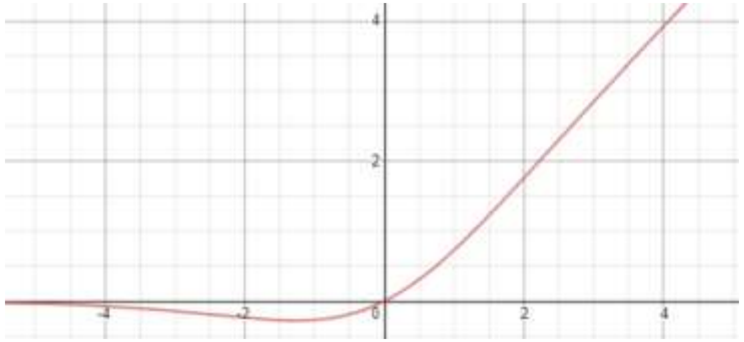
Summary: Activation function

- Recent progresses
  - Mish $f(x) = x \cdot \tanh(\varsigma(x))$, $\varsigma(x) = \ln(1+e^{\wedge}x)$,



https://arxiv.org/abs/1908.08681

  - Swish $f(x) = x * (1 + \exp(-x))^{-1}$



https://arxiv.org/abs/1710.05941

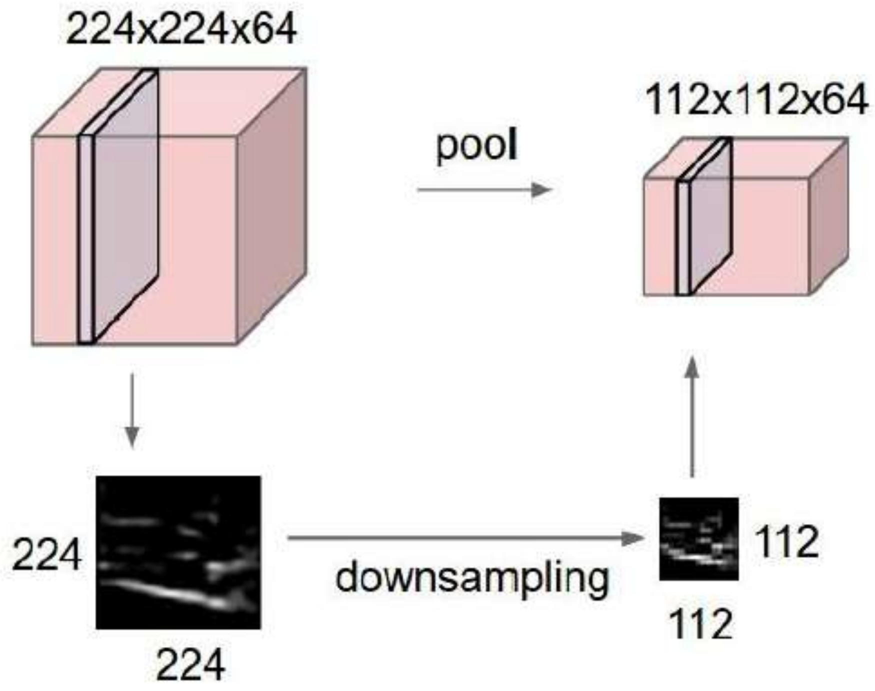
Outline

- Why Convolutional Neural Network (CNN)?
  - Motivation and overview
- What is the CNN?
  - Convolution layers & model complexity
  - Closer look at activation functions
  - Pooling layers & model complexity
  - Math properties
- Examples of CNNs

Acknowledgement: Roger Grosse@UofT & Feifei Li's cs231n notes

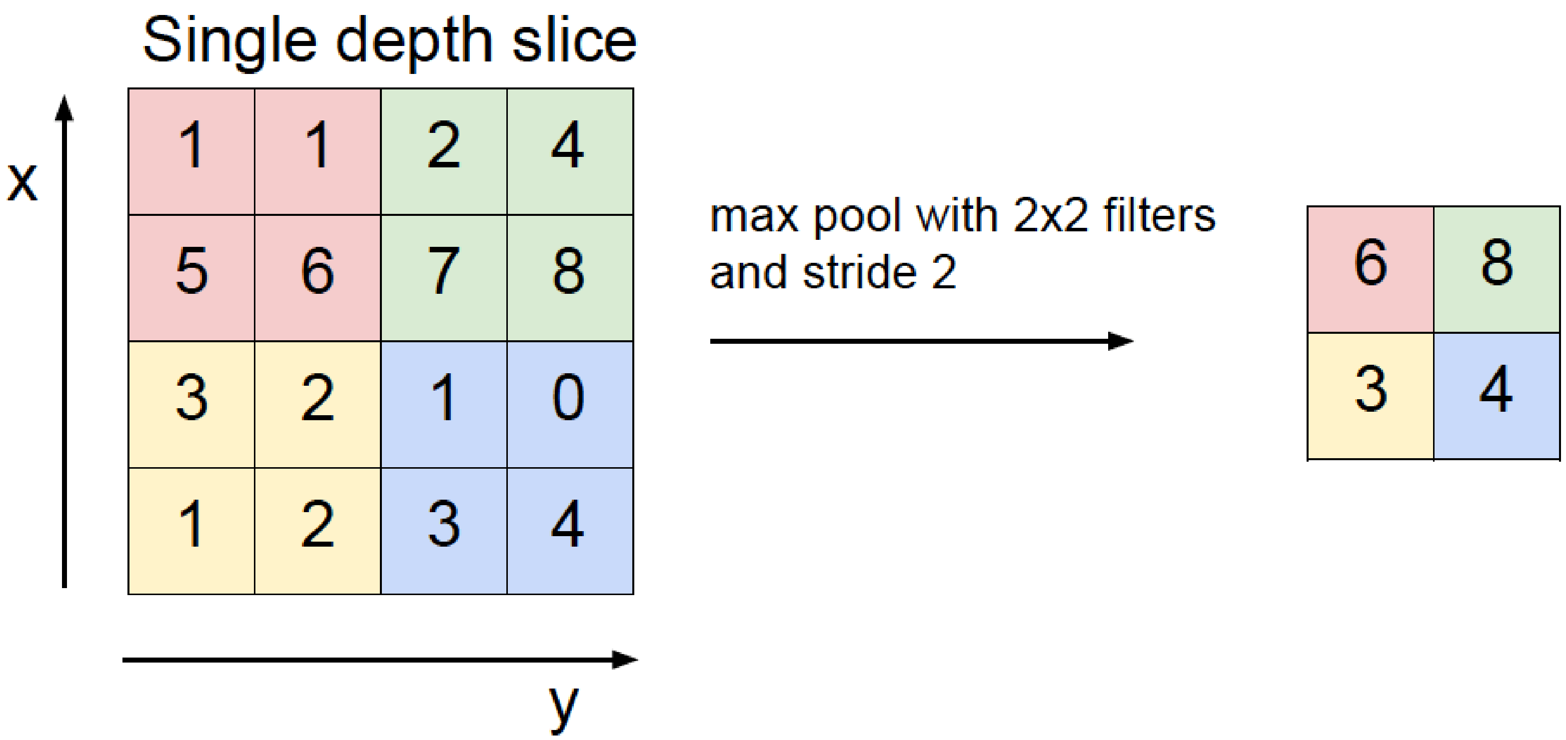
Pooling Layers

- Reducing the spatial size of the feature maps
  - Smaller representations
  - On each activation map independently
  - Low resolution means fewer details



Pooling Layers

- Example: max pooling
- Spatial invariance; no learnable parameters!



Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1
1	-2

$=$

-2		

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

*

0	1
1	-2

=

-2	-2	

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 1 \\ & & \\ & & \end{bmatrix}$$

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1
1	-2

$=$

-2	-2	1
0		

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 1 & 2 & 2 & 1 & 0 \\ \hline
0 & 2 & 4 & 4 & 2 & 0 \\ \hline
0 & 1 & 3 & 3 & 1 & 0 \\ \hline
0 & 1 & 2 & 3 & 1 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 0 \\ \hline
\end{array}
*
\begin{array}{|c|c|}
\hline
0 & 1 \\ \hline
1 & -2 \\ \hline
\end{array}
=
\begin{array}{|c|c|c|}
\hline
-2 & -2 & 1 \\ \hline
0 & 1 & 1 \\ \hline
1 & 2 & 0 \\ \hline
\end{array}
$$

- Reduces the dimensionality of the input to subsequent layers and thus, the number of weights to be learned

- Many relevant macro-features will tend to span large portions of the image, so taking strides with the convolution tends not to miss out on too much

Complexity of Pooling Layers

- Summary

  - 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

Outline

- Why Convolutional Neural Network (CNN)?
  - Motivation and overview
- What is the CNN?
  - Convolution layers & model complexity
  - Closer look at activation functions
  - Pooling layers & model complexity
  - Math properties
- Examples of CNNs

Acknowledgement: Roger Grosse@UofT & Feifei Li's cs231n notes

Math Properties of CNNs

上海科技大学 ShanghaiTech University

- What representations a CNN can capture in general?
- Consider a representation ϕ as an abstract function

$$\phi : \mathbf{x} \rightarrow \phi(\mathbf{x}) \in \mathbb{R}^d$$

- We want to look at how the representation changes upon transformations of input image.
  - Transformations represent the potential variations in the natural images
  - Translation, scale change, rotation, local deformation etc.

Math Properties of CNNs

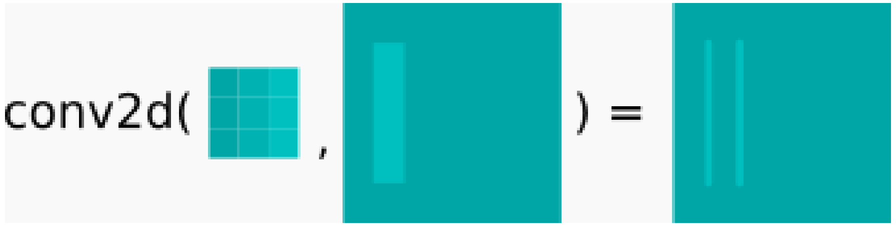
上海科技大学 ShanghaiTech University

- Two key properties of representations
  - Equivariance

A representation ϕ is equivariant with a transformation g if the transformation can be transferred to the representation output.

$$\exists \text{ a map } M_g : \mathbb{R}^d \rightarrow \mathbb{R}^d \text{ such that:}$$
$$\forall \mathbf{x} \in \mathcal{X} : \phi(g\mathbf{x}) \approx M_g \phi(\mathbf{x})$$

  - Example: convolution w.r.t. translation



Math Properties of CNNs

- Two key properties of representations
  - Invariance

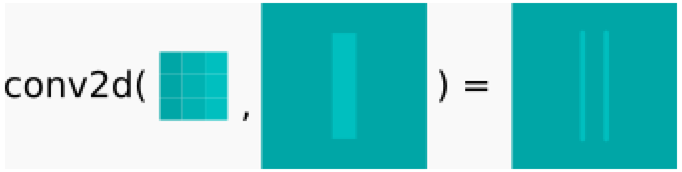
    A representation ϕ is invariant with a transformation g if the transformation has no effect on the representation output.

$$\forall \mathbf{x} \in \mathcal{X} : \phi(g\mathbf{x}) \approx \phi(\mathbf{x})$$

  - Example: convolution+pooling+FC w.r.t. translation

Math Properties of CNNs

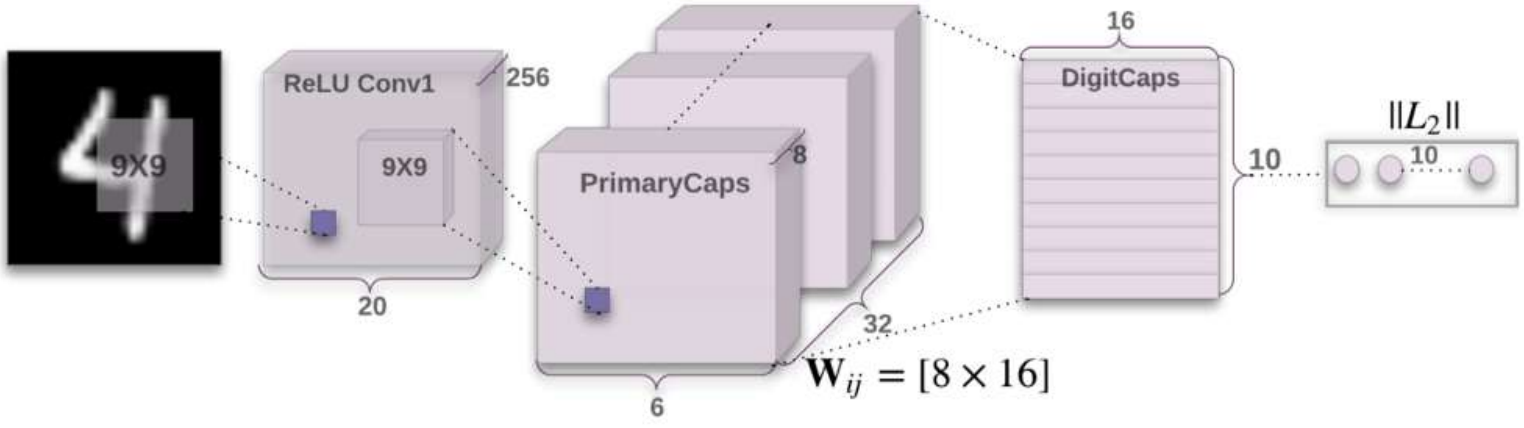
- Recent results on convolution layers
  - Convolutions are equivariant to translation
  - Convolutions are not equivariant to other isometries of the sampling lattice, e.g., rotation



  - What if a CNN learns rotated copies of the same filter?
    - The stack of feature maps is equivariant to rotation.

Math Properties of CNNs

- Recent results on convolution layers
  - Ordinary CNNs can be generalized to Group Equivariant Networks (Cohen and Welling ICML'16, Kondor and Trivedi ICML'18)
    - Redefining the convolution and pooling operations
    - Equivariant to more general transformation from some group *G*
  - Replacing pooling by other network designs
    - Capsule network (Sabour et al, 2017) https://arxiv.org/abs/1710.09829



Outline

上海科技大学
ShanghaiTech University

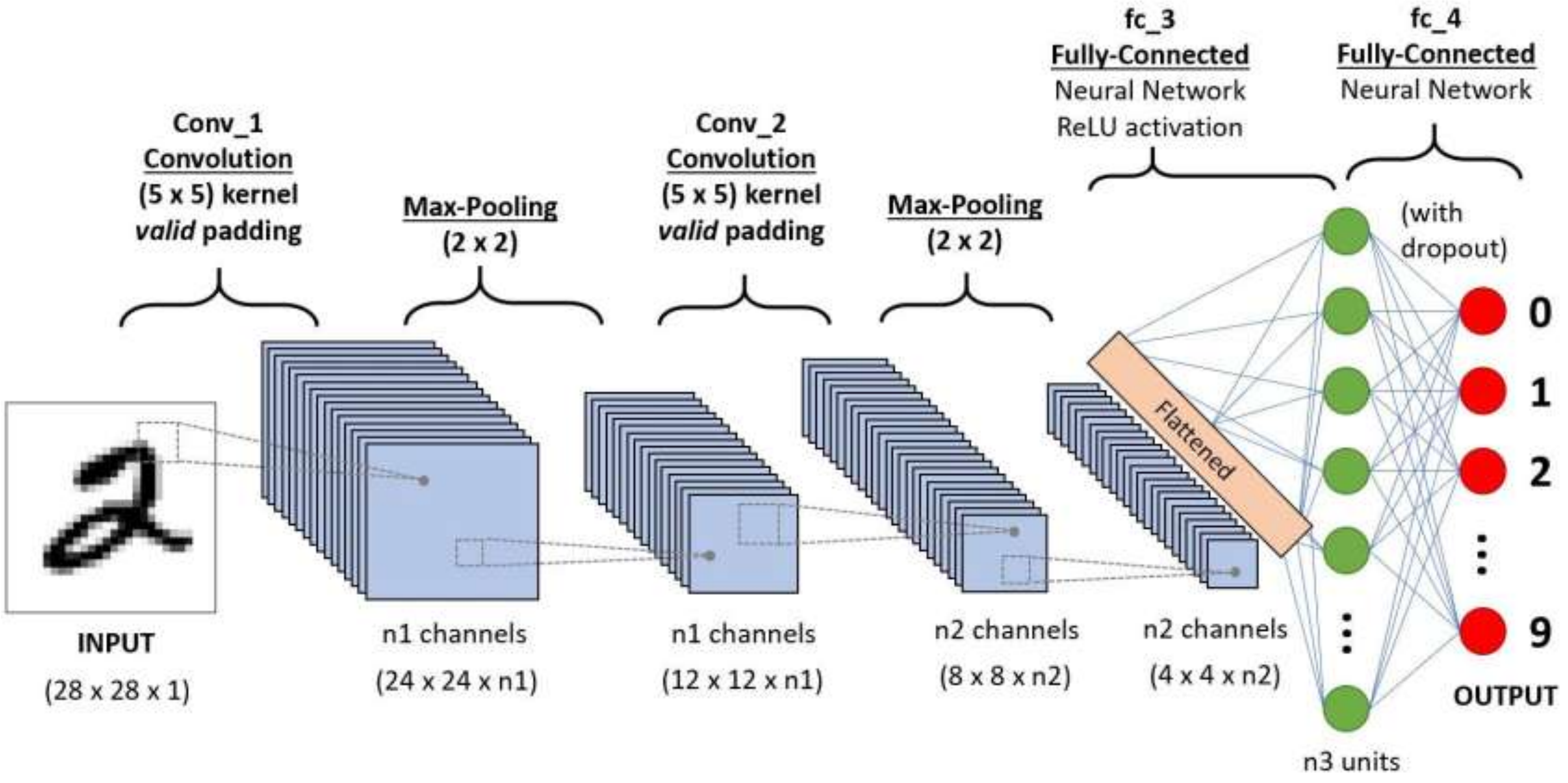
- Why Convolutional Neural Network (CNN)?
  - Motivation and overview
- What is the CNN?
  - Convolution layers & model complexity
  - Closer look at activation functions
  - Pooling layers & model complexity
  - Math properties
- Examples of CNNs

Acknowledgement: Roger Grosse@UofT & Feifei Li's cs231n notes

LeNet-5

上海科技大学 ShanghaiTech University

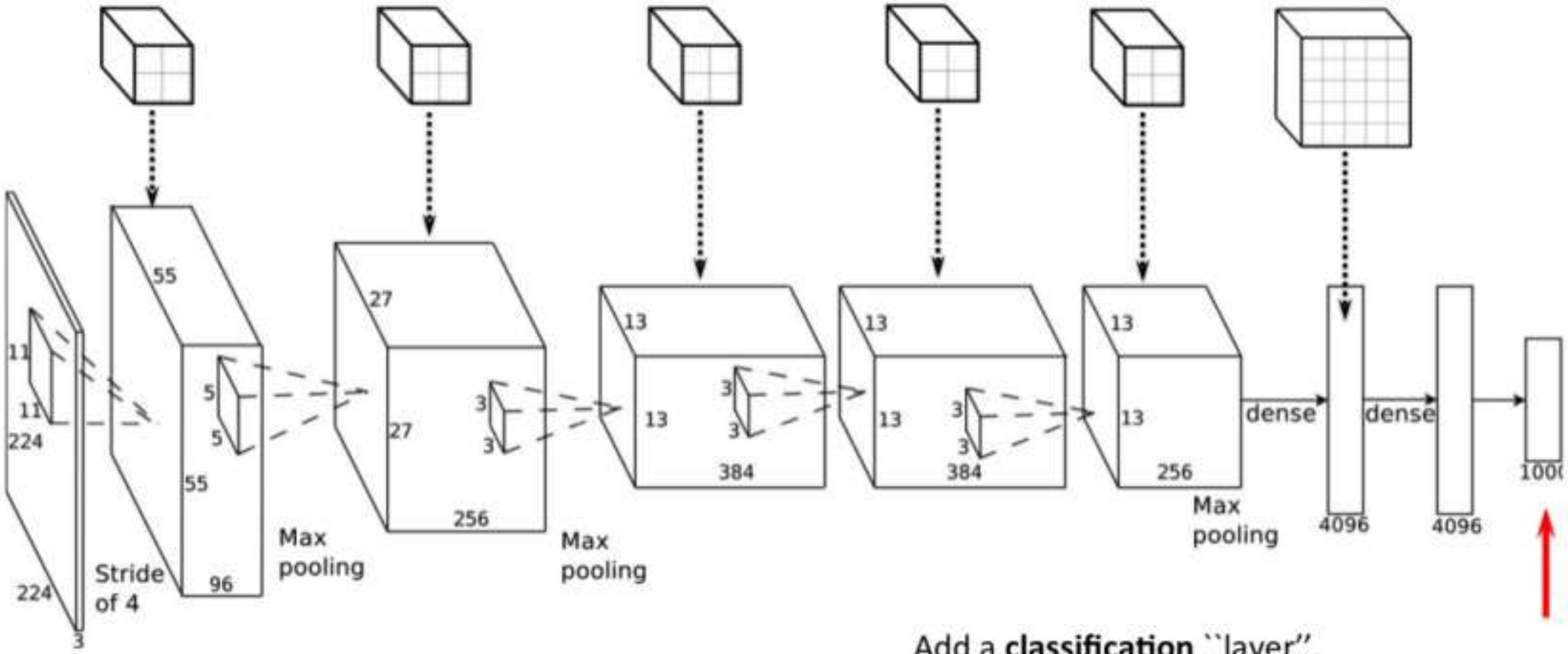
- Handwritten digit recognition



AlexNet

上海科技大学
ShanghaiTech University

- Deeper network structure



Add a **classification** ``layer''.

For an input image, the value in a particular dimension of this vector tells you the probability of the corresponding object class.