SE125 Machine Learning

Convolutional Neural Networks

Yue Ding

School of Software, Shanghai Jiao Tong University
dingyue@sjtu.edu.cn

Convolutional Neural Networks

- 课程难度:

- 掌握程度:

References and Acknowledgement

- Stanford CS231n: Convolutional Neural Networks for Visual Recognition
- PIERIAN DATA, Convolutional Neural Networks

Image Classification

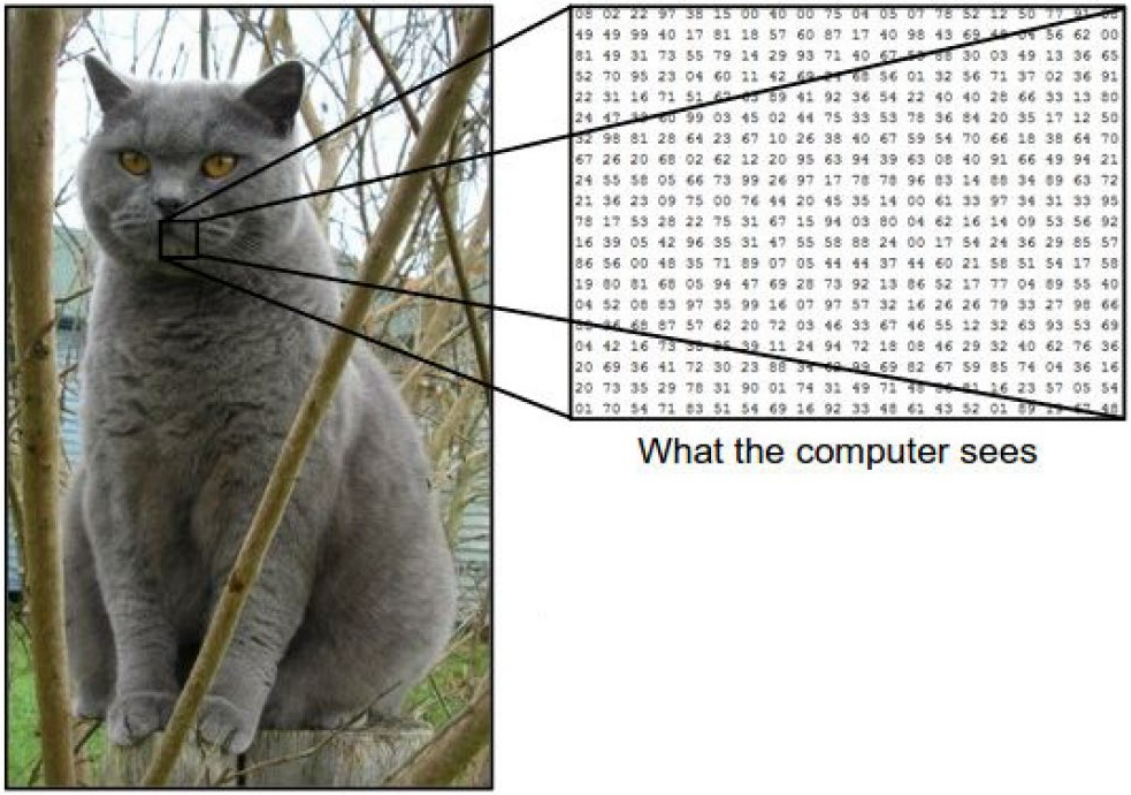
(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

cat

The Problem: *Semantic Gap*

- Images are represented as 3D arrays of numbers, with integers between [0, 255].

- E.g. 300 x 100 x 3 (3 for 3 color channels RGB)



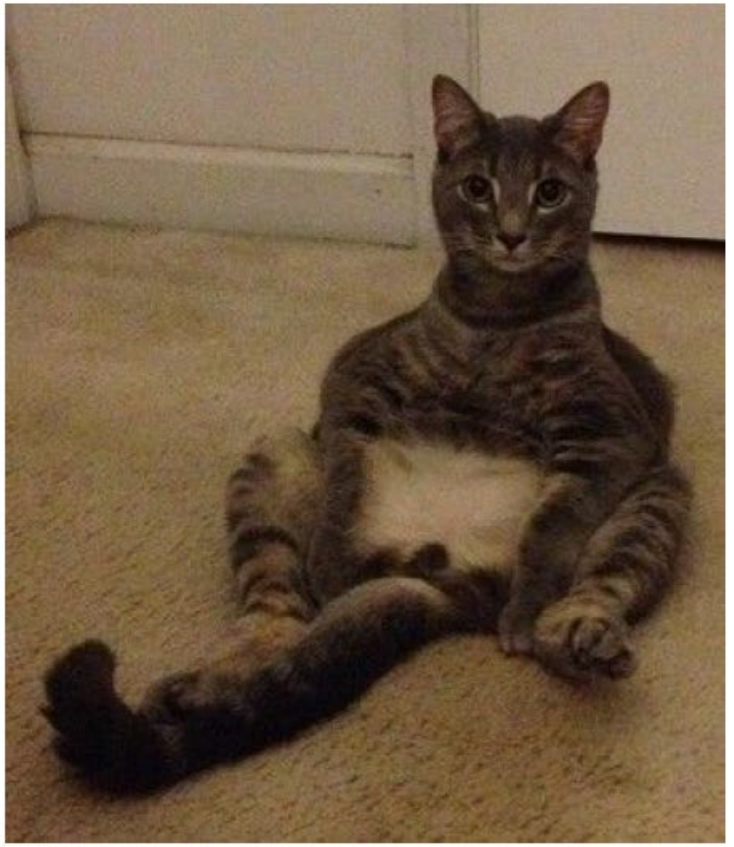
What the computer sees

Challenges

- Occlusion

Challenges

- Deformation



Challenges

- Background clutter

Intraclass Variation

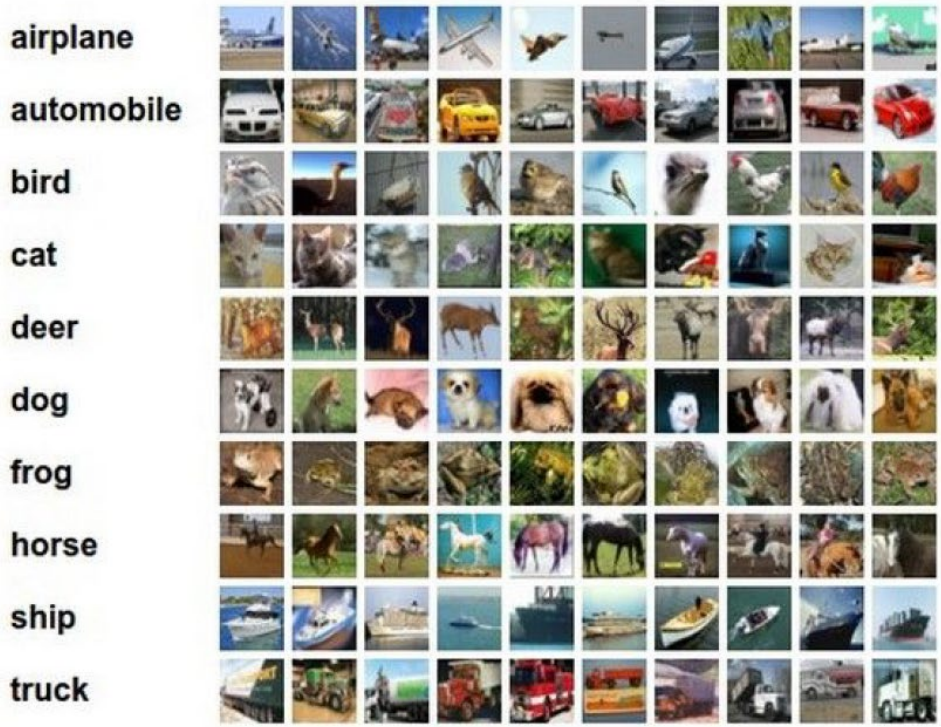
- Intraclass variation

Data-driven Approach:

- 1. Collect a dataset of images and labels
- 2. Use Machine Learning to train an image classifier
- 3. Evaluate the classifier on a withheld set of test images

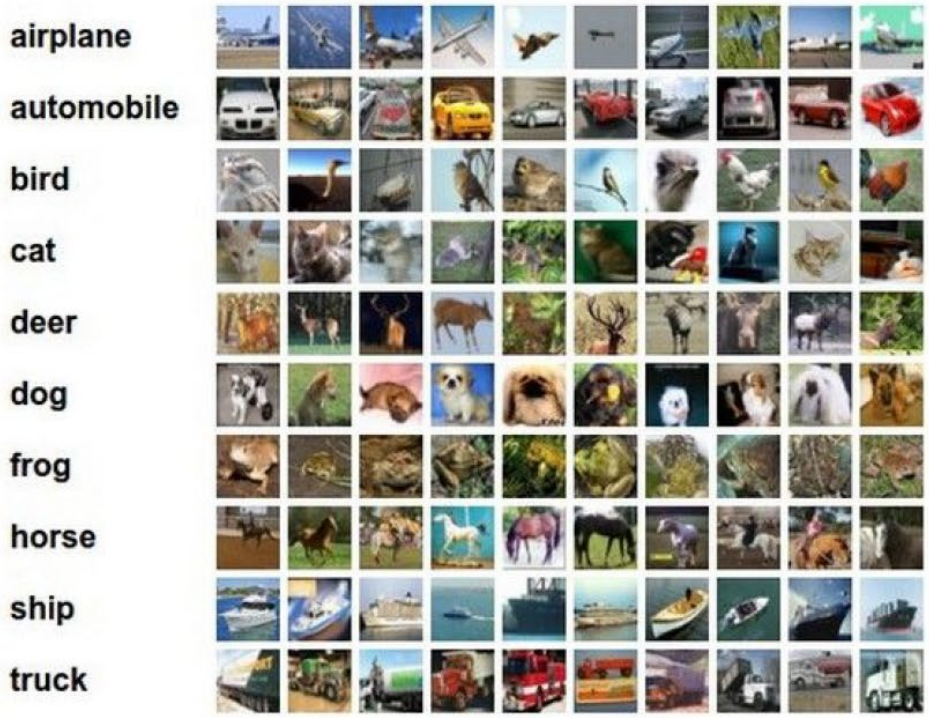
Nearest Neighbor Classifier

Example dataset: **CIFAR-10**
10 labels
50,000 training images, each image is tiny: 32x32
10,000 test images.



Nearest Neighbor Classifier

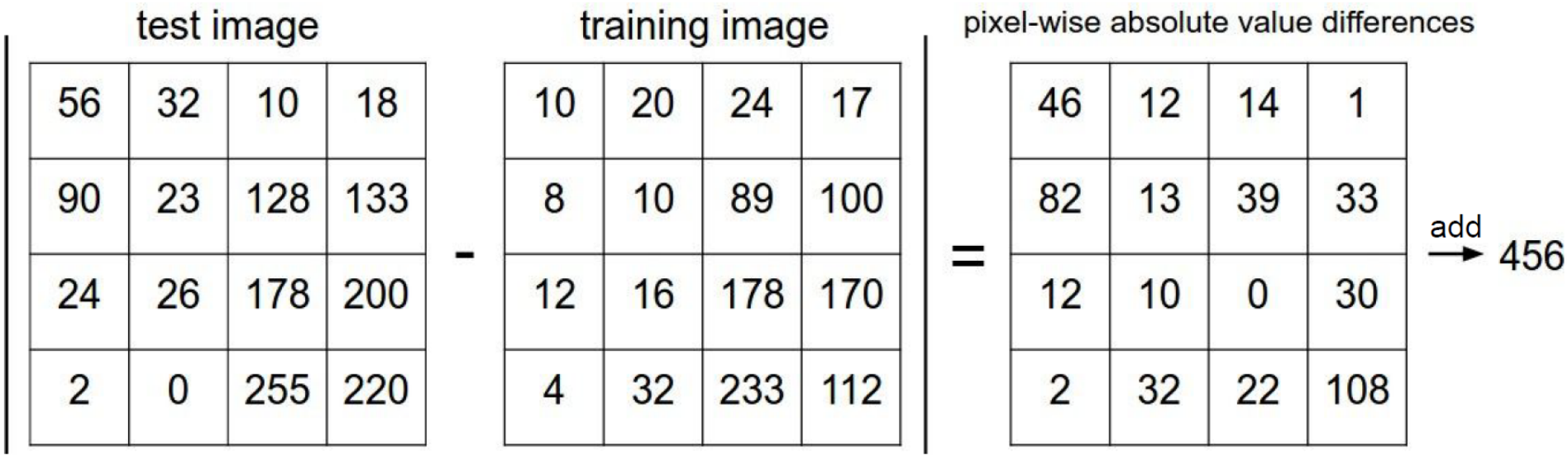
Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.



For every test image (first column), examples of nearest neighbors in rows

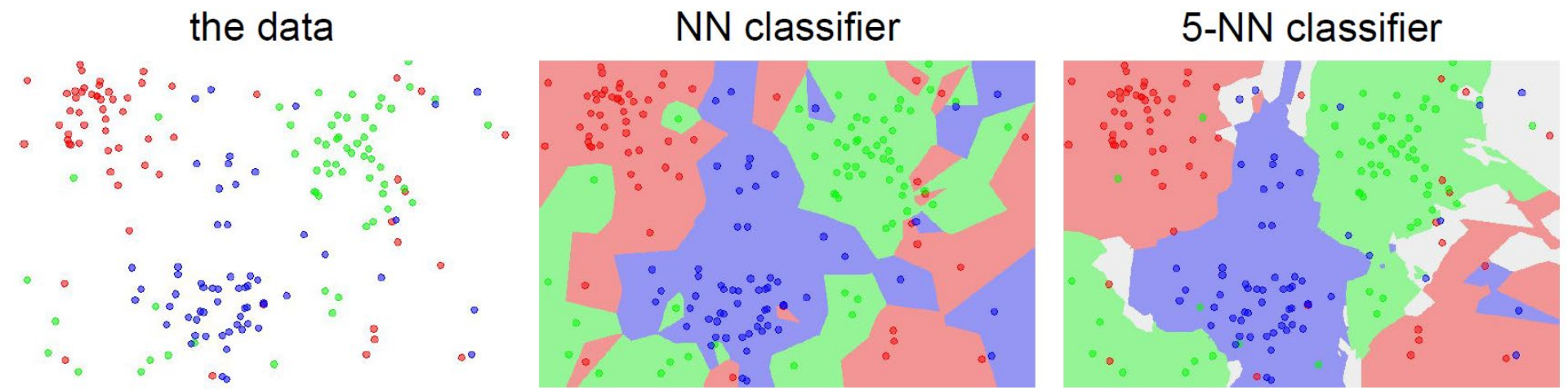
How do we compare the images? What is the **distance metric**?

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$



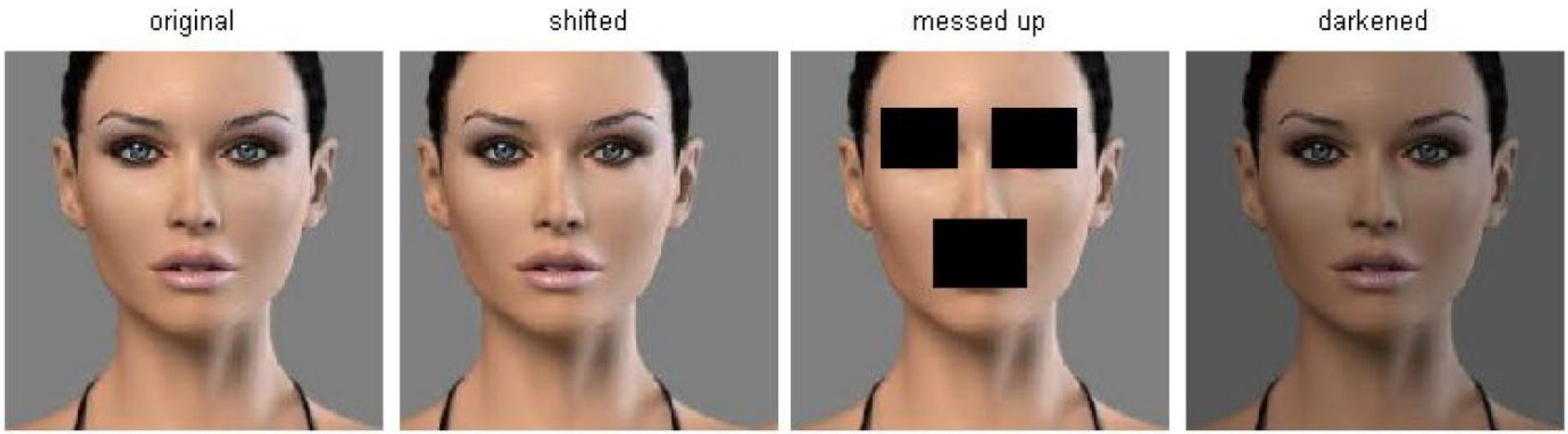
k-Nearest Neighbor

- Find the k nearest images, have them vote on the label



k-Nearest Neighbor

- k-Nearest Neighbor on images **never used**
  - terrible performance at test time
  - distance metrics on level of whole images can be very unintuitive



(all 3 images have same L2 distance to the one on the left)

Linear Classification

- Parametric approach: **Linear classifier**

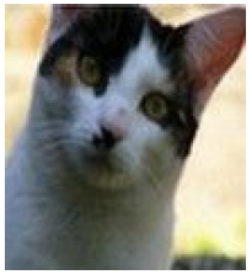


image parameters

$f(\mathbf{x}, \mathbf{W})$

10 numbers, indicating class scores

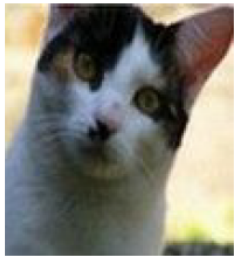
[32x32x3]
array of numbers 0...1
(3072 numbers total)

Linear Classification

- Parametric approach: **Linear classifier**

$$f(x, W) = Wx$$

10x1 (f(x, W)) 10x3072 (W) 3072x1 (x) (+b) 10x1



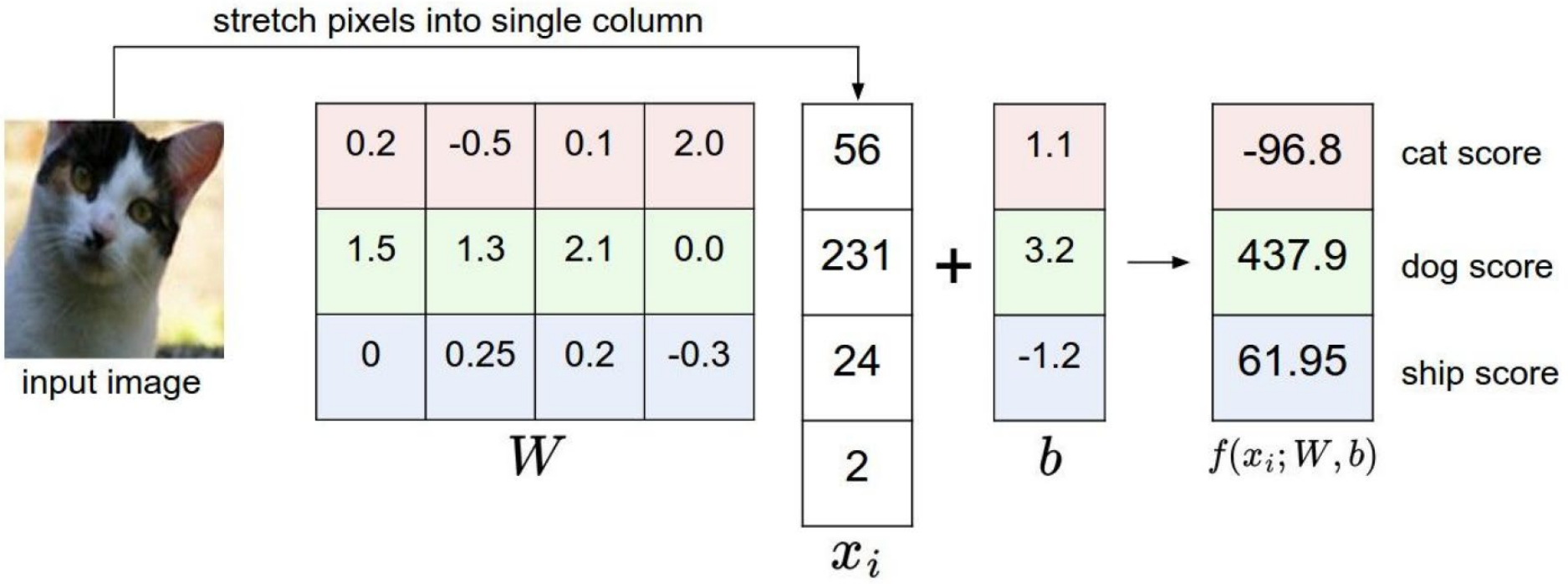
[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

parameters, or “weights”

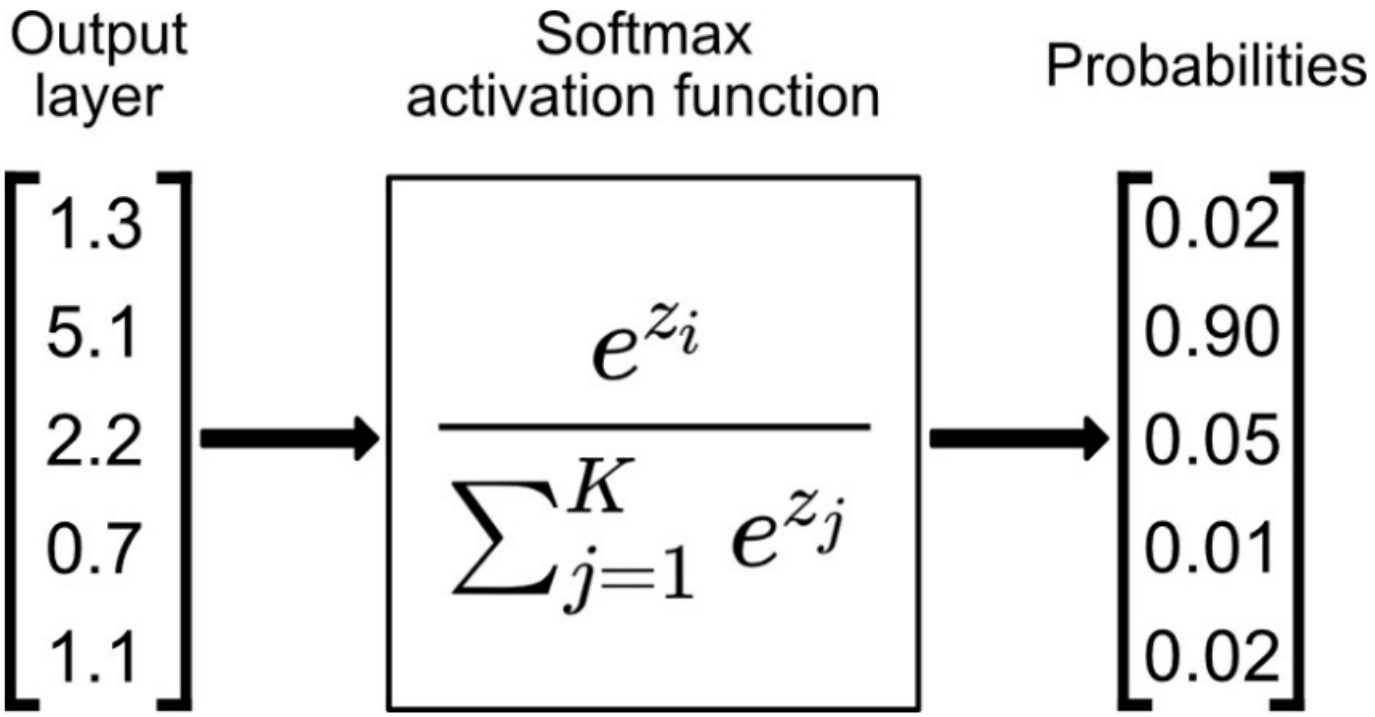
Linear Classification

- Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

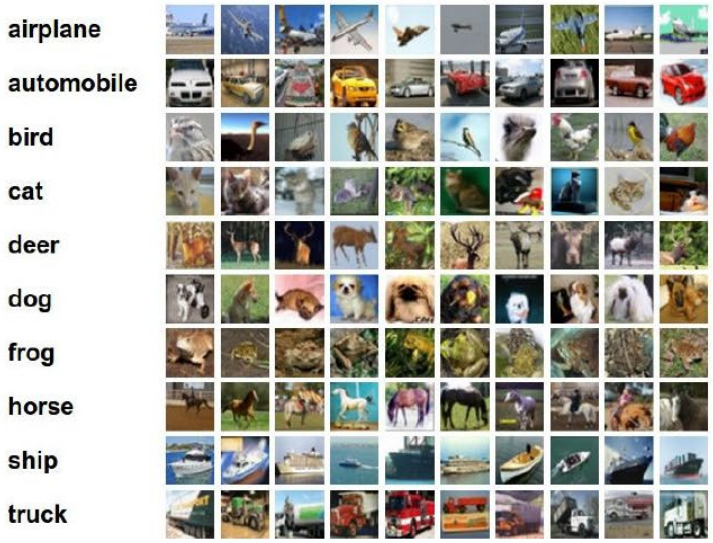


Linear Classification

- The softmax function



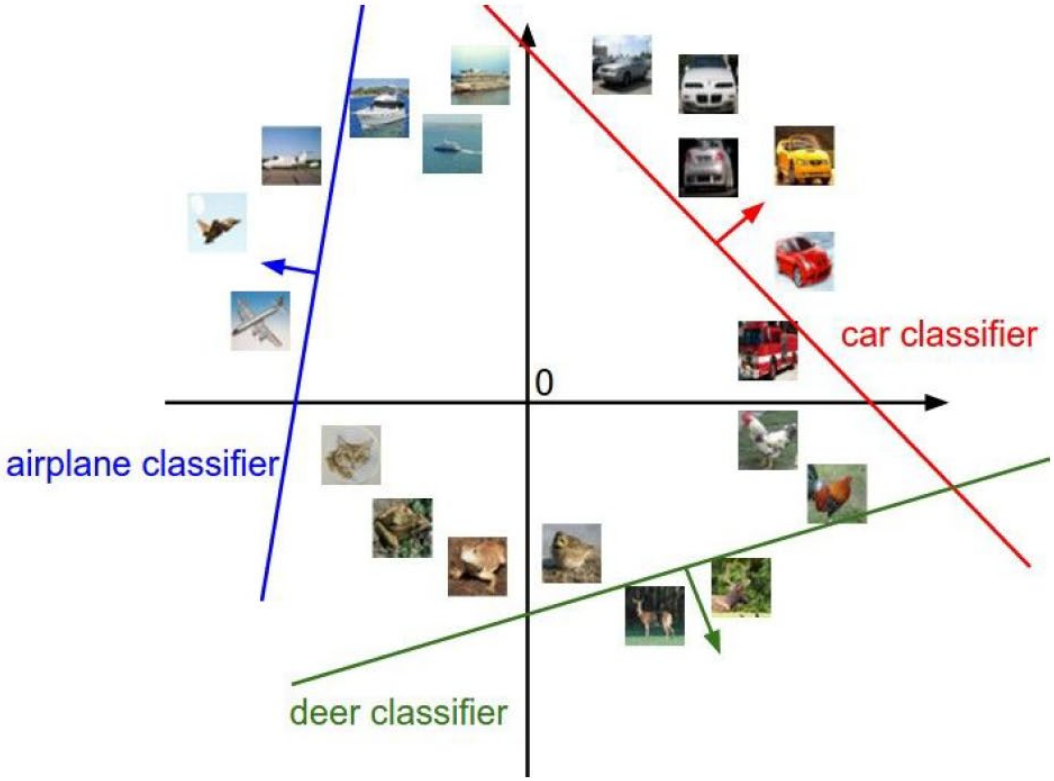
Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

Interpreting a Linear Classifier

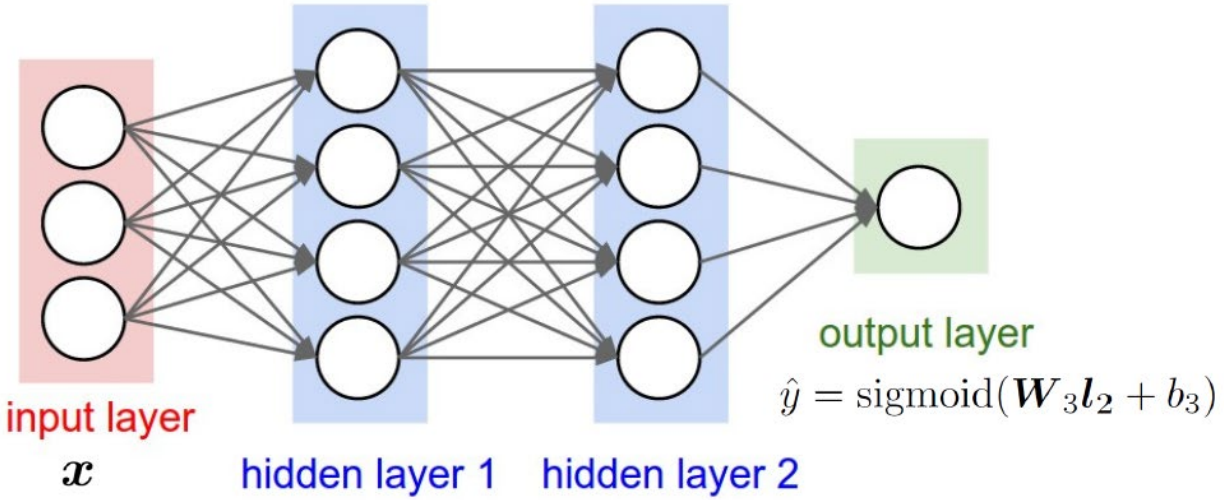


$$f(x_i, W, b) = Wx_i + b$$

[32x32x3]
array of numbers 0...1
(3072 numbers total)

MLP as a Classifier

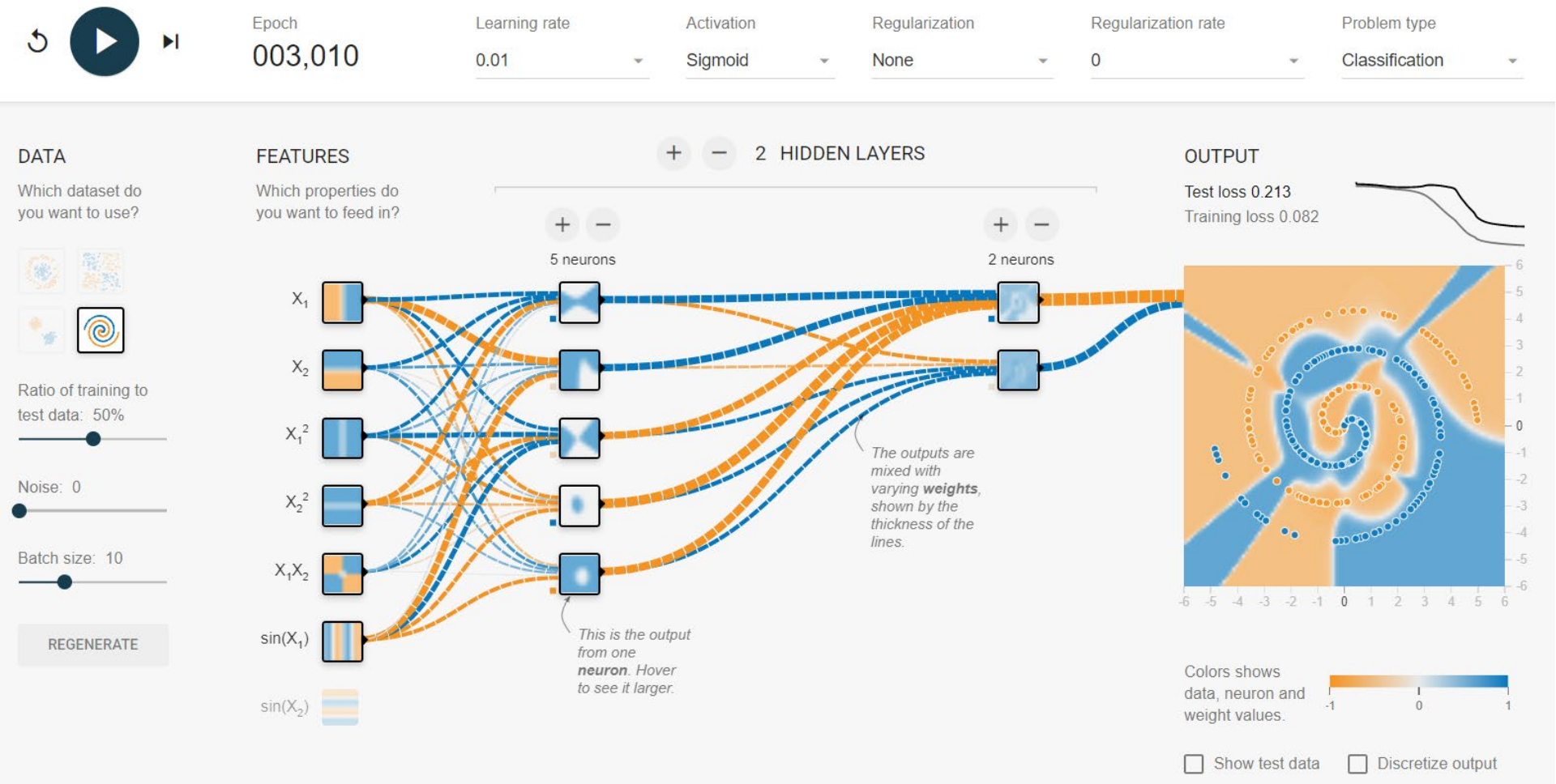
- An MLP with one hidden layer can approximate any nonlinear function of the input given sufficiently many hidden units.



$$\boldsymbol{l}_1 = \tanh(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \quad \boldsymbol{l}_2 = \tanh(\boldsymbol{W}_2\boldsymbol{l}_1 + \boldsymbol{b}_2)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$

A Demo from Google



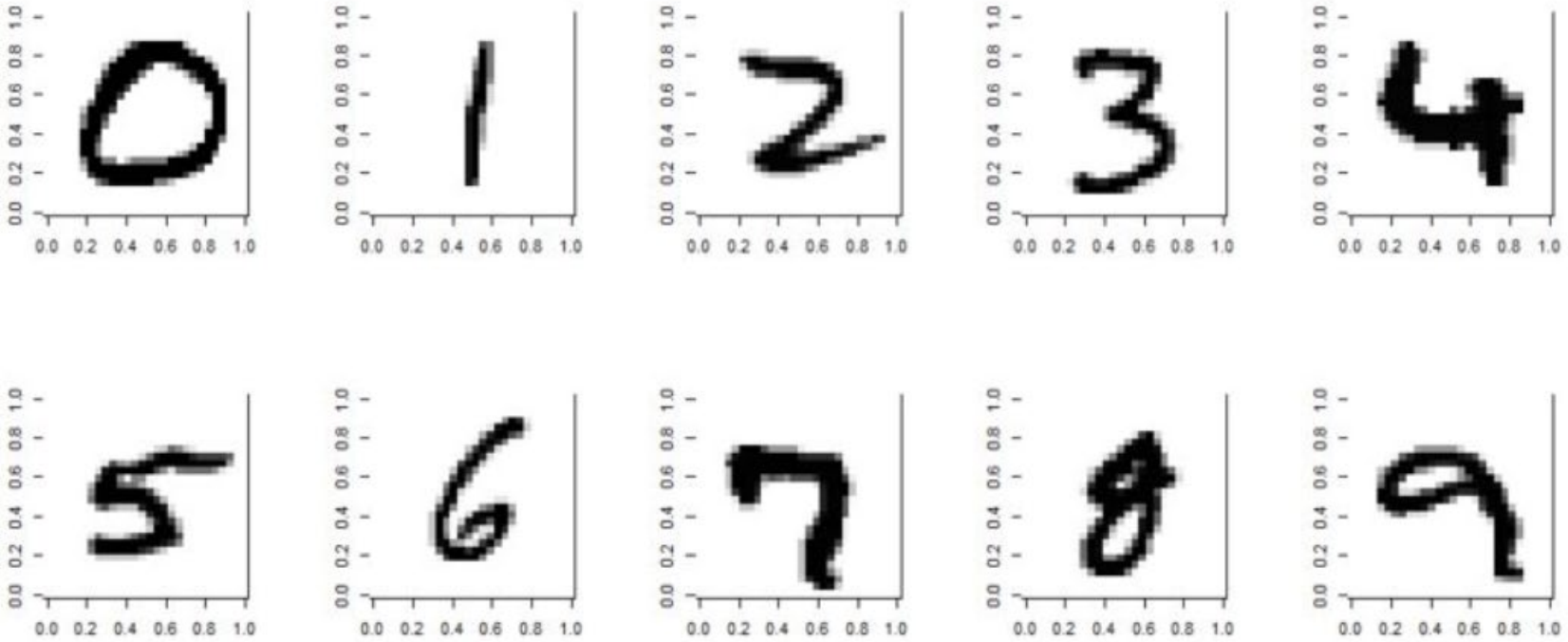
http://playground.tensorflow.org/

MNIST Data

- A classic data set in Deep Learning
- Some basics about it
  - 55,000 training images
  - 5,000 validation images
  - 10,000 test images
- The MNIST data set contains handwritten single digits from 0 to 9

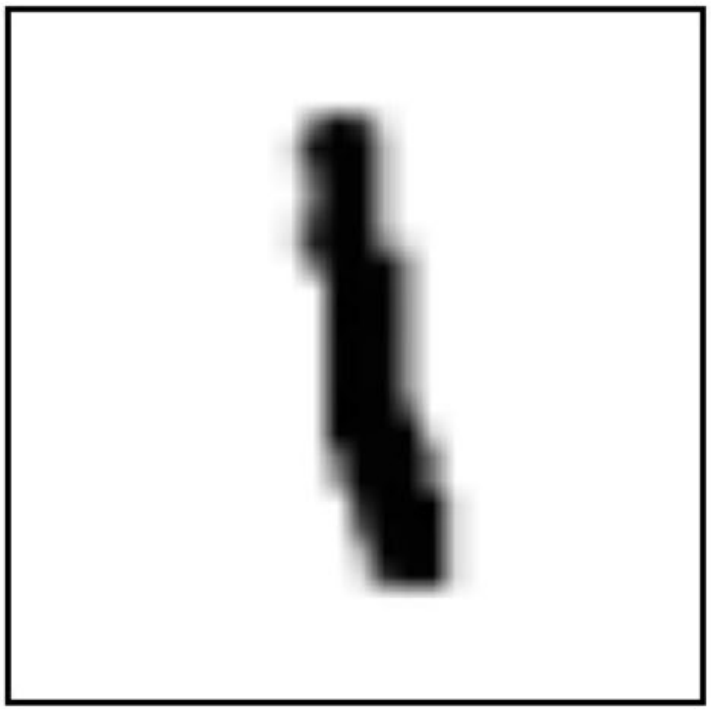
MNIST Data

- A single digit image can be represented as an array



MNIST Data

- Specifically, 28 by 28 pixels

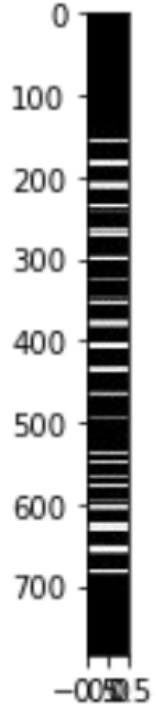


≈

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

MNIST Data

- We can flatten this array to a 1-D vector of 784 numbers. Either (784,1) or (1,784) is fine, as long as the dimensions are consistent.



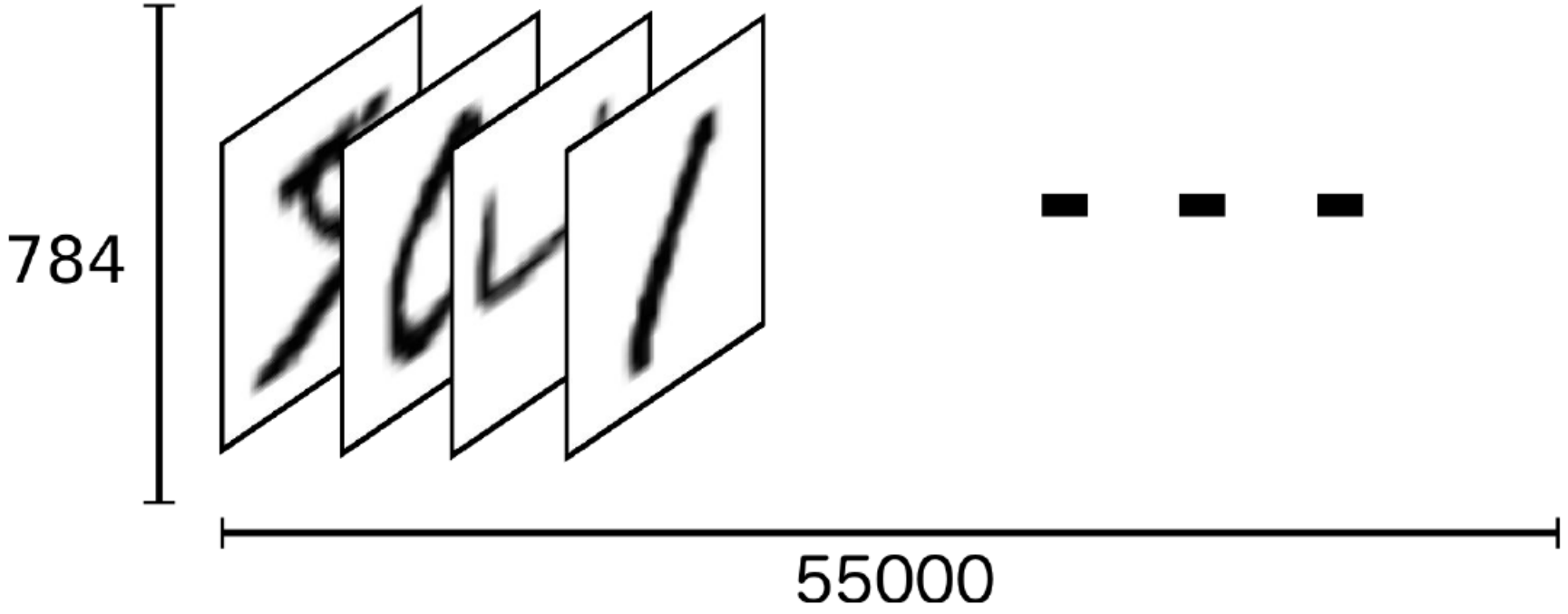
MNIST Data

- Flattening out the image ends up removing some of the 2-D information, such as the relationship of a pixel to its neighboring pixels.

- For now, we'll ignore this, but come back to it later when we discuss CNN in depth!

MNIST Data

- We can think of the entire group of the 55,000 images as a tensor (an n-dimensional array)



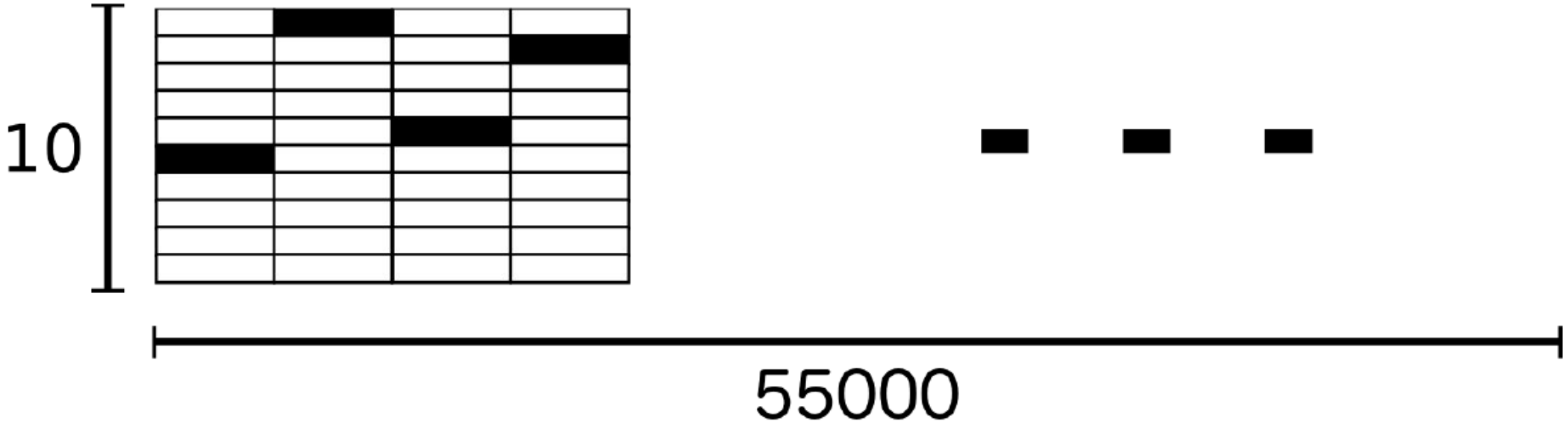
MNIST Data

- For the labels we'll use One-Hot Encoding.
- This means that instead of having labels such as "One", "Two", etc… we'll have a single array for each image.
- For example, 4 would have this label array:

[0,0,0,0,1,0,0,0,0,0]

MNIST Data

- As a result, the labels for the training data ends up being a large 2-d array (10,55000):



MLP as a Classifier

- Let’s see the code.

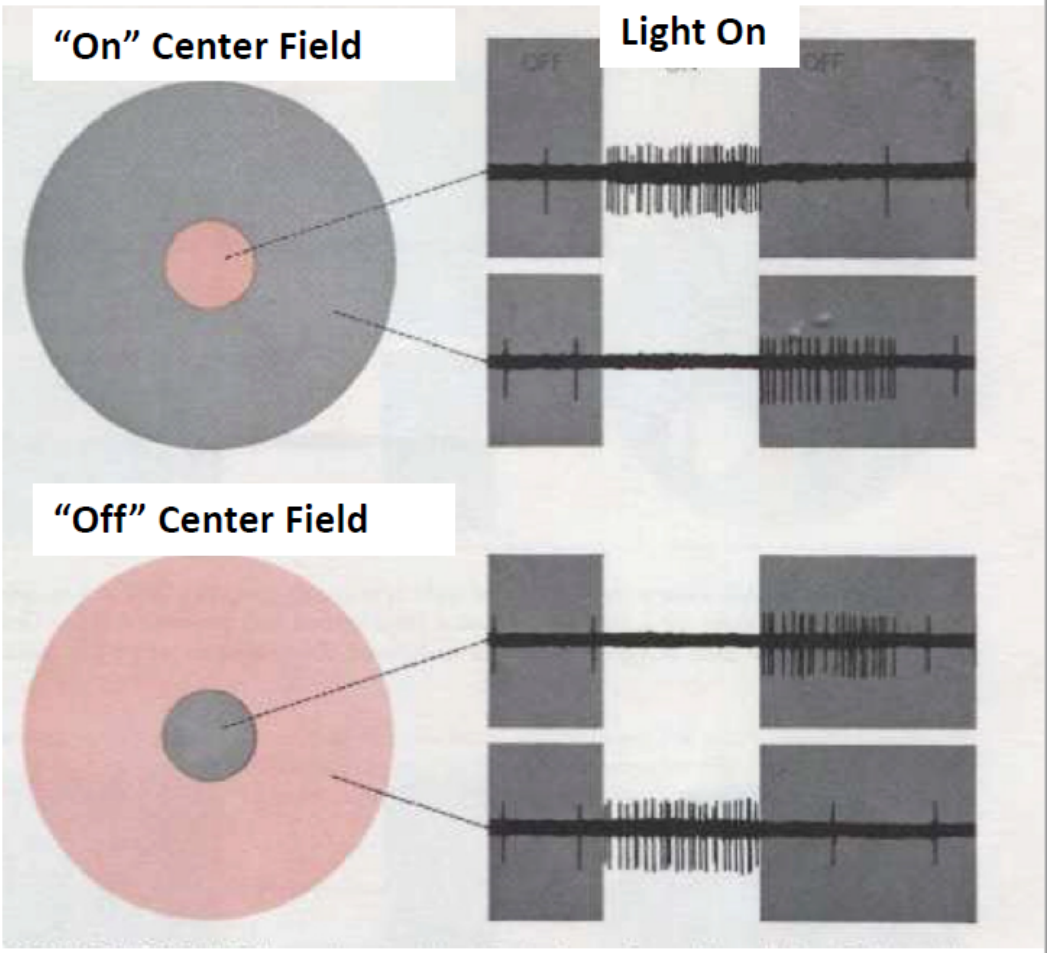
Convolutional Neural Networks

- Let's explore a much better approach using Convolutional Neural Networks.
- Just like the simple perceptron, CNNs also have their origins in biological research.
  - Hubel and Wiesel studied the structure of the visual cortex in mammals, winning a Nobel Prize in 1981.

Receptive Field

- Receptive field: Neurons in the retina respond to light stimulus in restricted regions of the visual field

- Animal experiments on receptive fields of two retinal ganglion cells
  - Fields are circular areas of the retina
  - The cell (upper part) responds when the center is illuminated and the surround is darkened.
  - The cell (lower part) responds when the center is darkened and the surround is illuminated.
  - Both cells give on- and off-responses when both center and surround are illuminated, but neither response is as strong as when only center or surround is illuminated

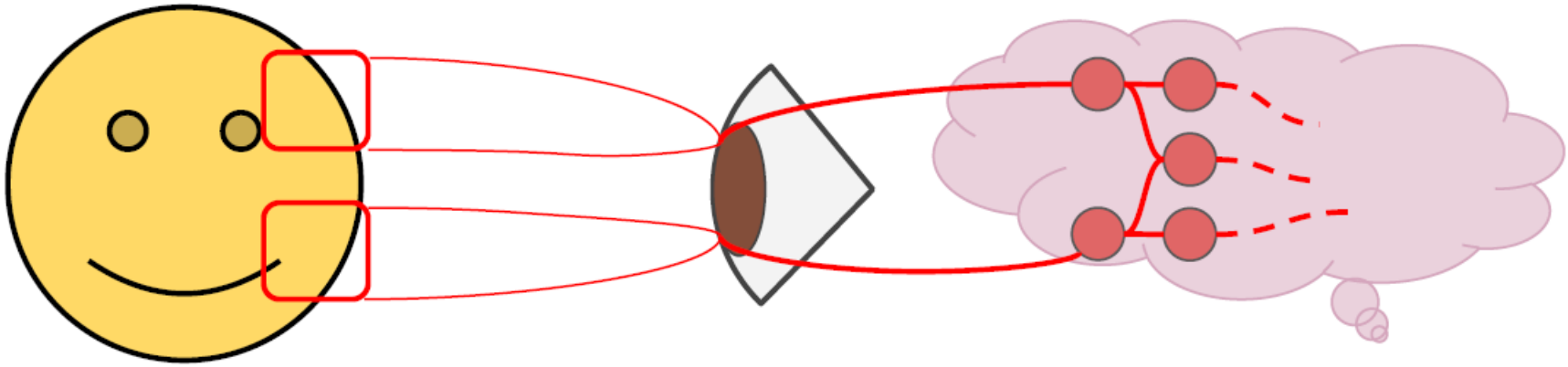


Contributed by Hubel and Wiesel for the studies

Hubel D.H. : The Visual Cortex of the Brain. Sci Amer 209:54-62, 1963

Receptive Field

- Their research revealed that neurons in the visual cortex had a small local receptive field.

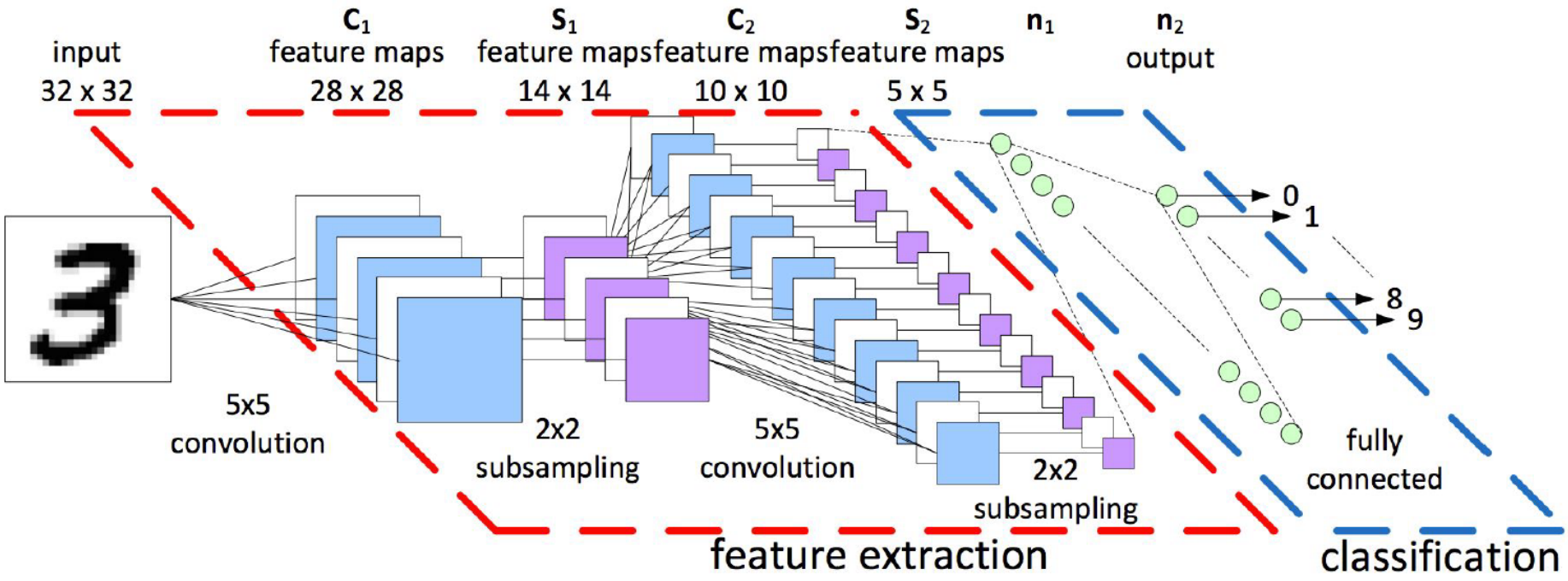


Convolutional Neural Networks

- This idea then inspired an ANN architecture that would become CNN
- Famously implemented in the 1998 paper by Yann LeCun et al.
- The LeNet-5 architecture was first used to classify the MNIST data set.

Convolutional Neural Networks

- When learning about CNNs you'll often see a diagram like this:



Convolutional Neural Networks

- Let’s break down the different aspects of a CNN :
  - Tensors
  - DNN vs CNN
  - Convolutions and Filters
  - Padding
  - Pooling
  - Flatten

Tensors

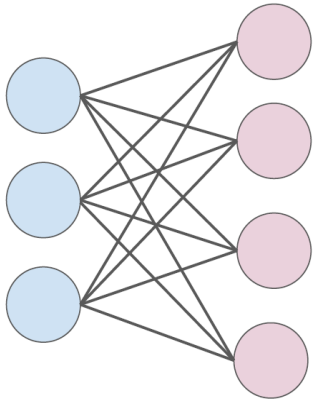
- Recall that Tensors are N-Dimensional Arrays that we build up to:
  - Scalar – 3
  - Vector - [3,4,5]
  - Matrix - [[3,4] , [5,6] , [7,8]]
  - Tensor - [[[1, 2] , [3, 4]], [[5, 6] , [7, 8]]]

Tensors

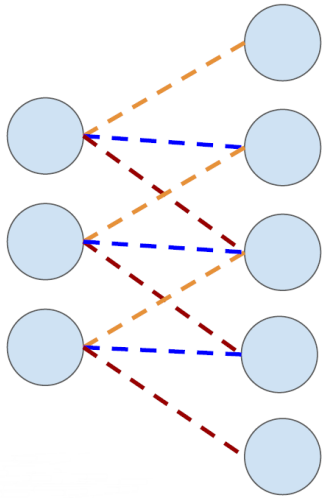
- Tensors make it very convenient to feed in sets of images into our model - (I,H,W,C)
  - I : Images
  - H: Height of Image in Pixels
  - W: Width of Image in Pixels
  - C: Color Channels: 1-Grayscale, 3-RGB

DNN vs CNN

Densely Connected layer



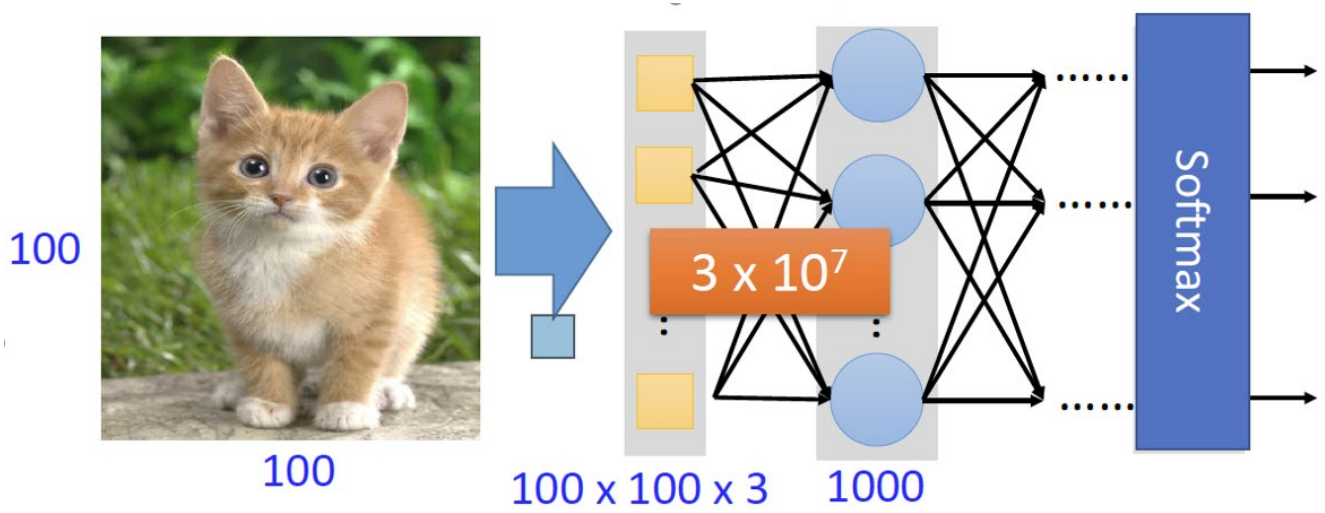
Convolutional Layer



- Each unit is connected to a smaller number of nearby units in next layer

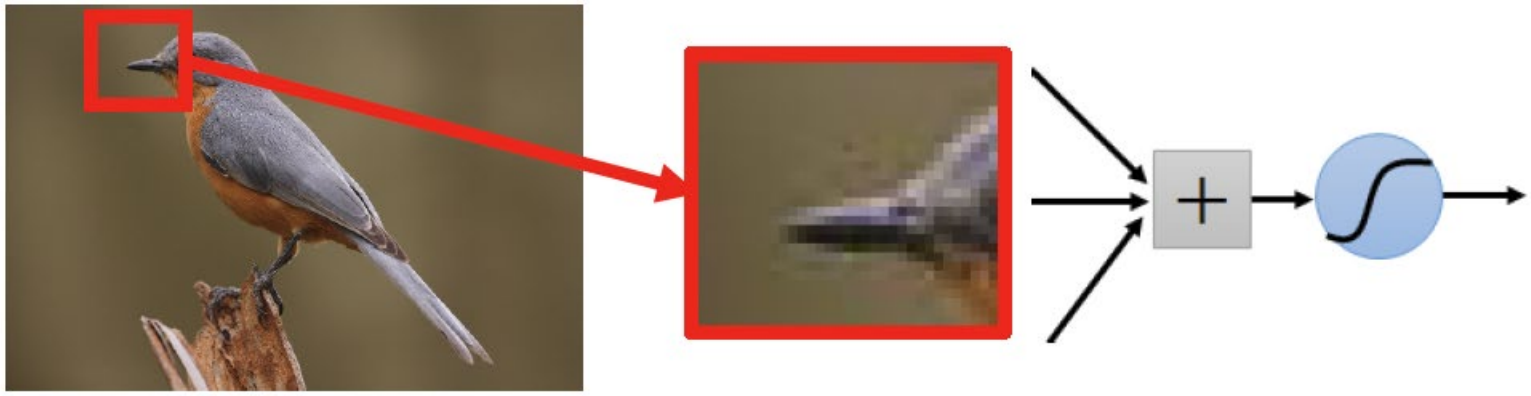
DNN vs CNN

- So why bother with a CNN instead of a DNN?
  - The MNIST dataset was 28 by 28 pixels (784 total)
  - But most images are at least 256 by 256 or greater, (<56k total)!
  - This leads to too many parameters, unscalable to new images.



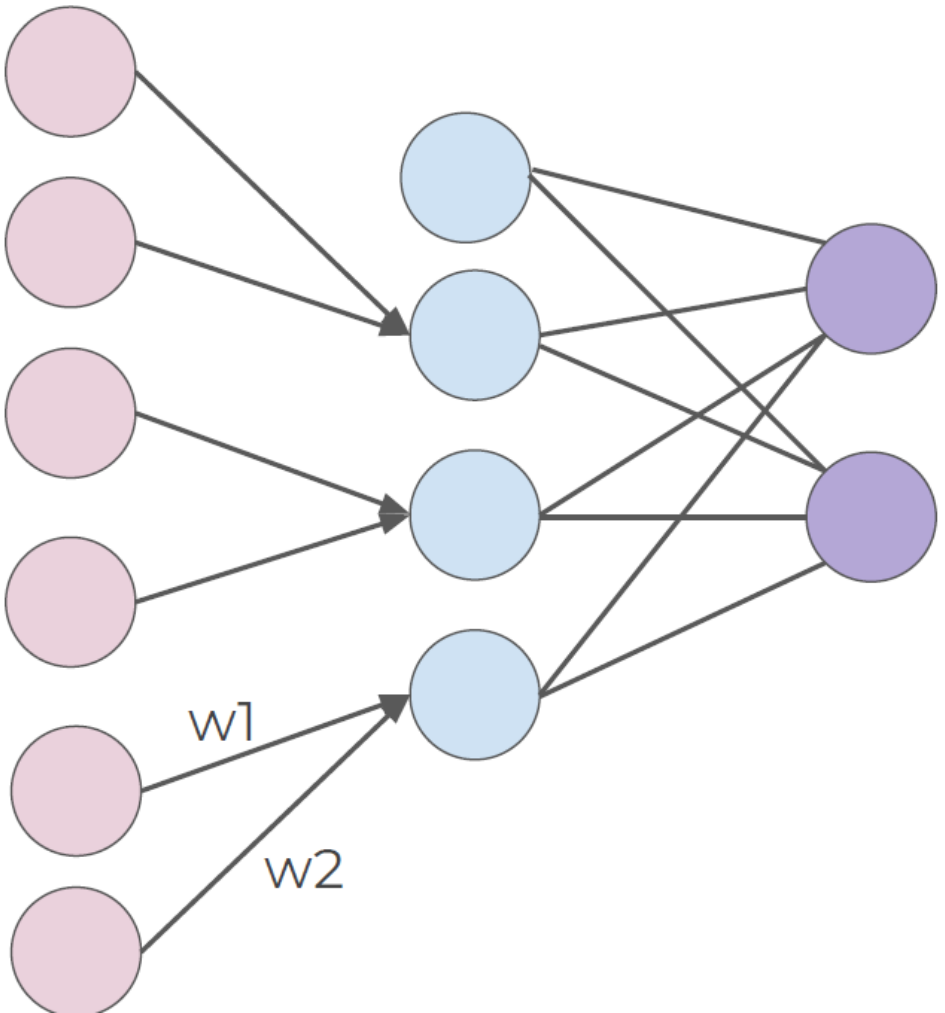
DNN vs CNN

- Convolutions also have a major advantage for image processing, where pixels nearby to each other are much more correlated to each other for image detection.
  - Some patterns are much smaller than the whole image, a neuron does not have to see the whole image to discover the pattern.



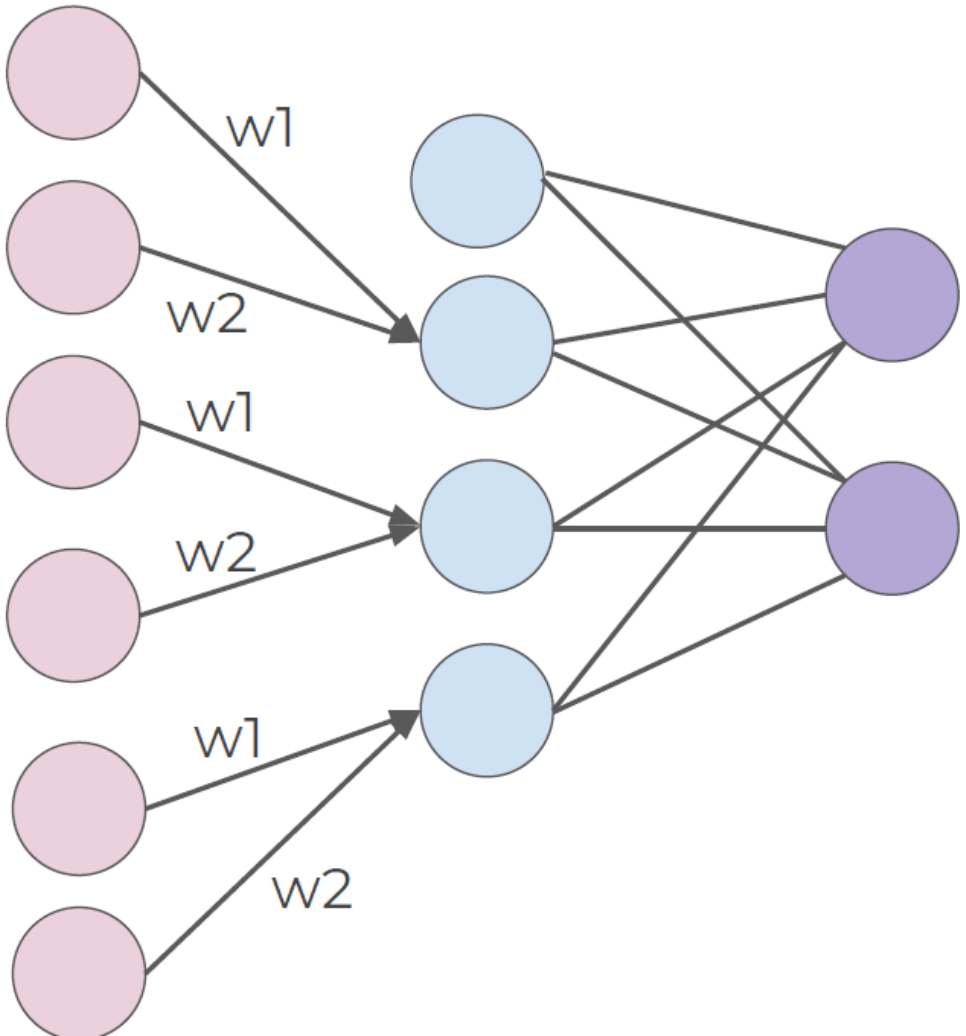
Convolutions and Filters

- 1-D Convolution
  - We can treat these weights as a filter.



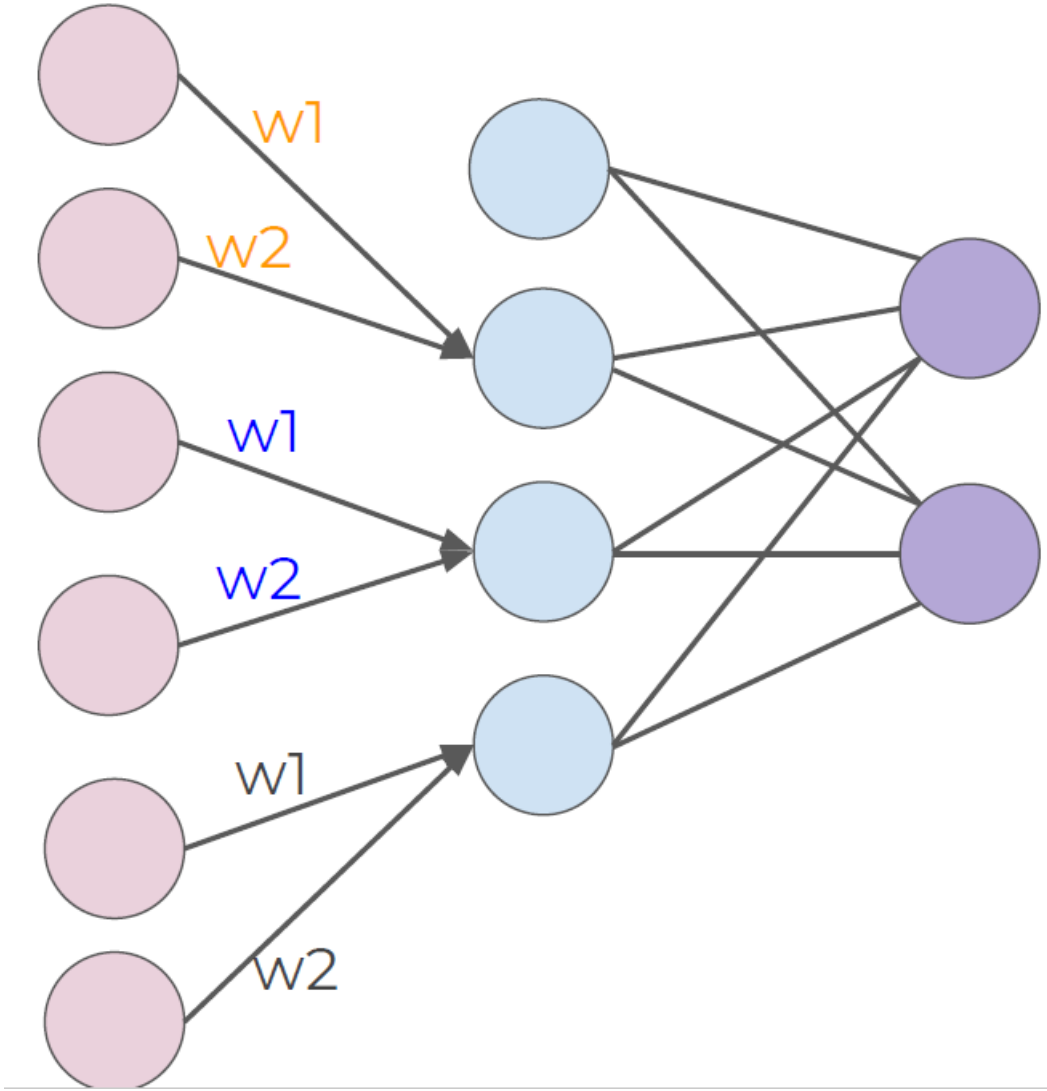
Convolutions and filters

- We now have a set of weights that can act as a filter.
- We can then expand this idea to multiple filters.



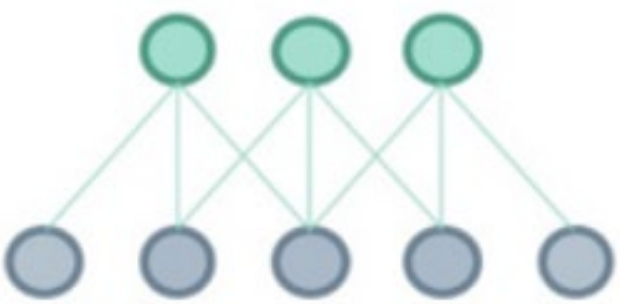
Convolutions and Filters

- 1-D Convolution
  - Filters: 1
  - Filter Size: 2
  - **Stride: 2**

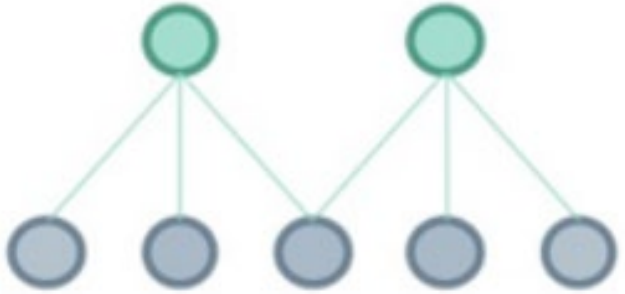


Convolutions and Filters

- **Stride (S)** with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 then the filters jump 2 pixels at a time.



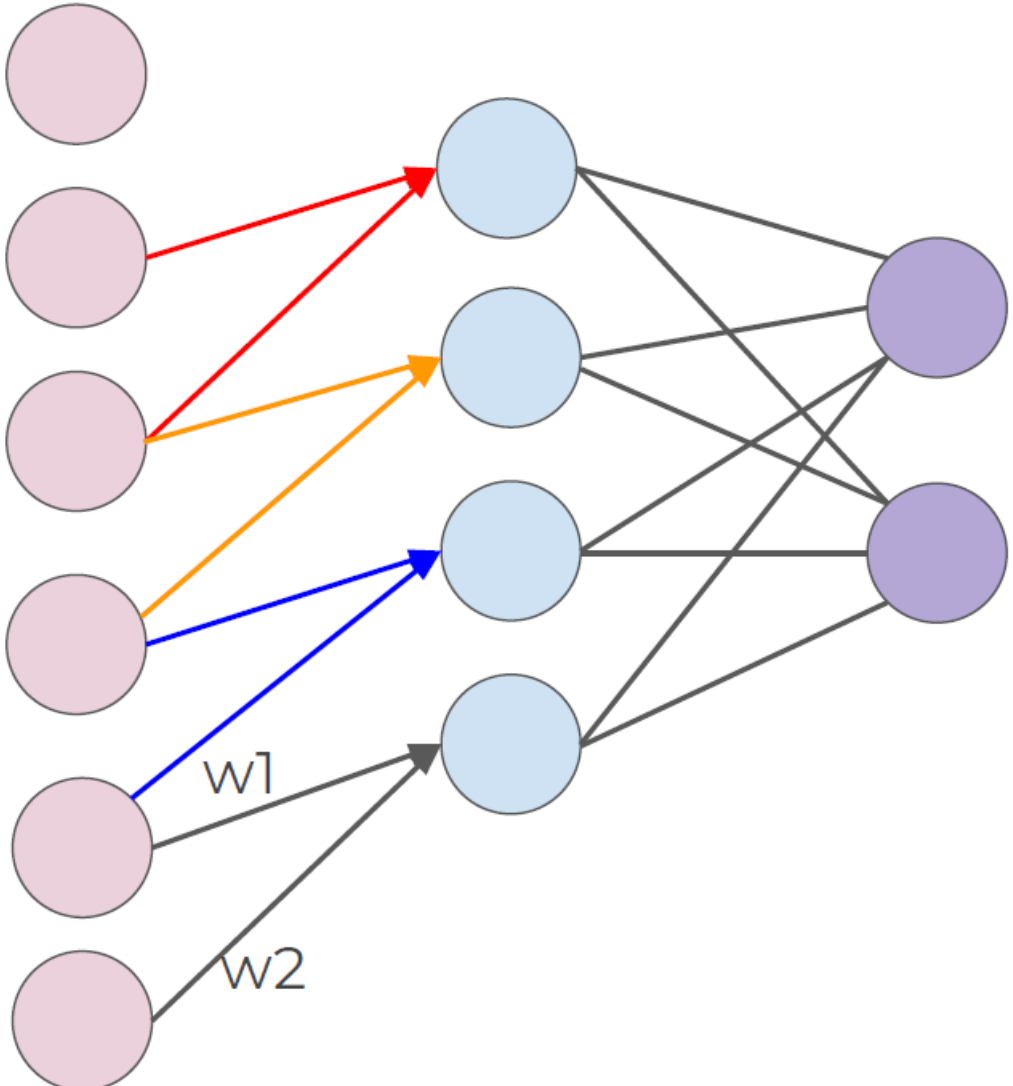
Stride = 1



Stride = 2

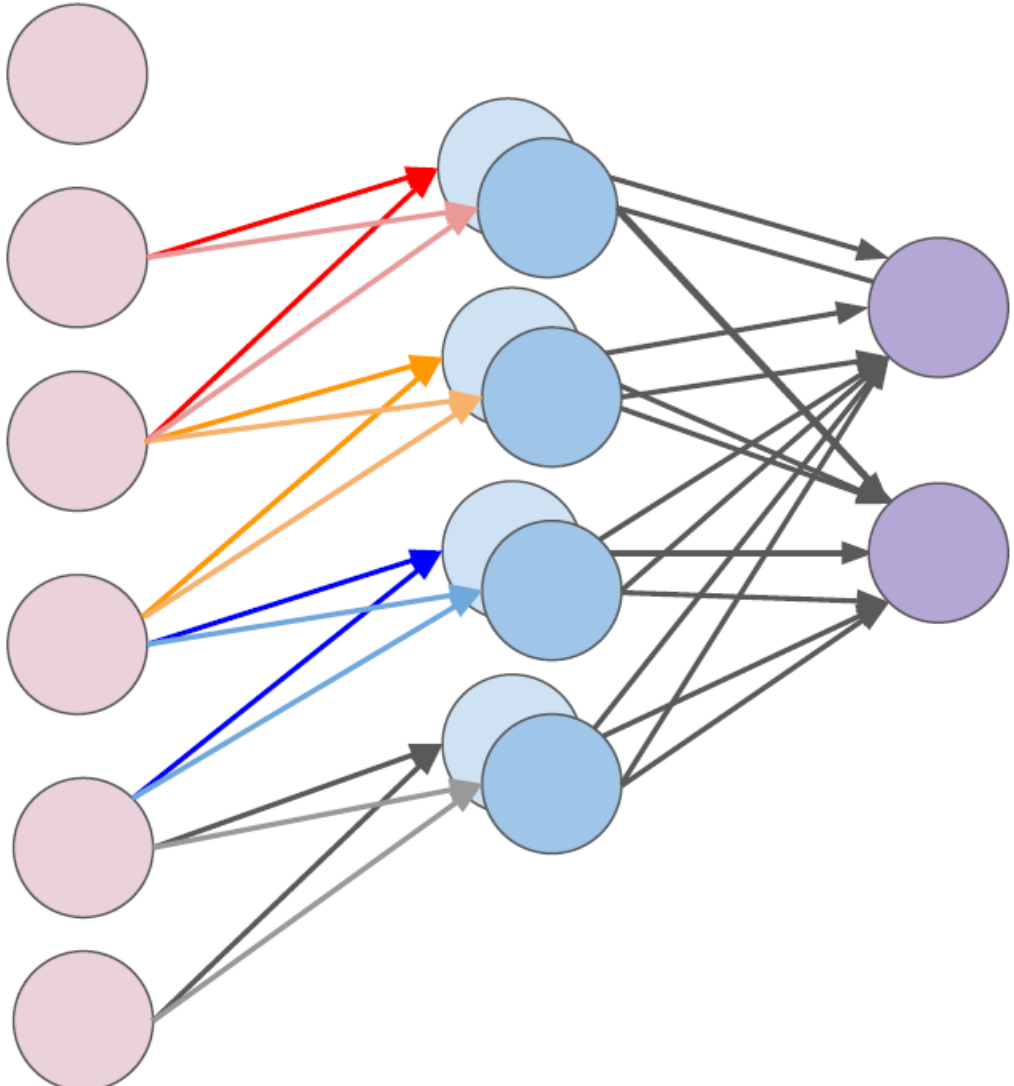
Convolutions and Filters

- 1-D Convolution
  - Filters: 1
  - Filter Size: 2
  - Stride: 1



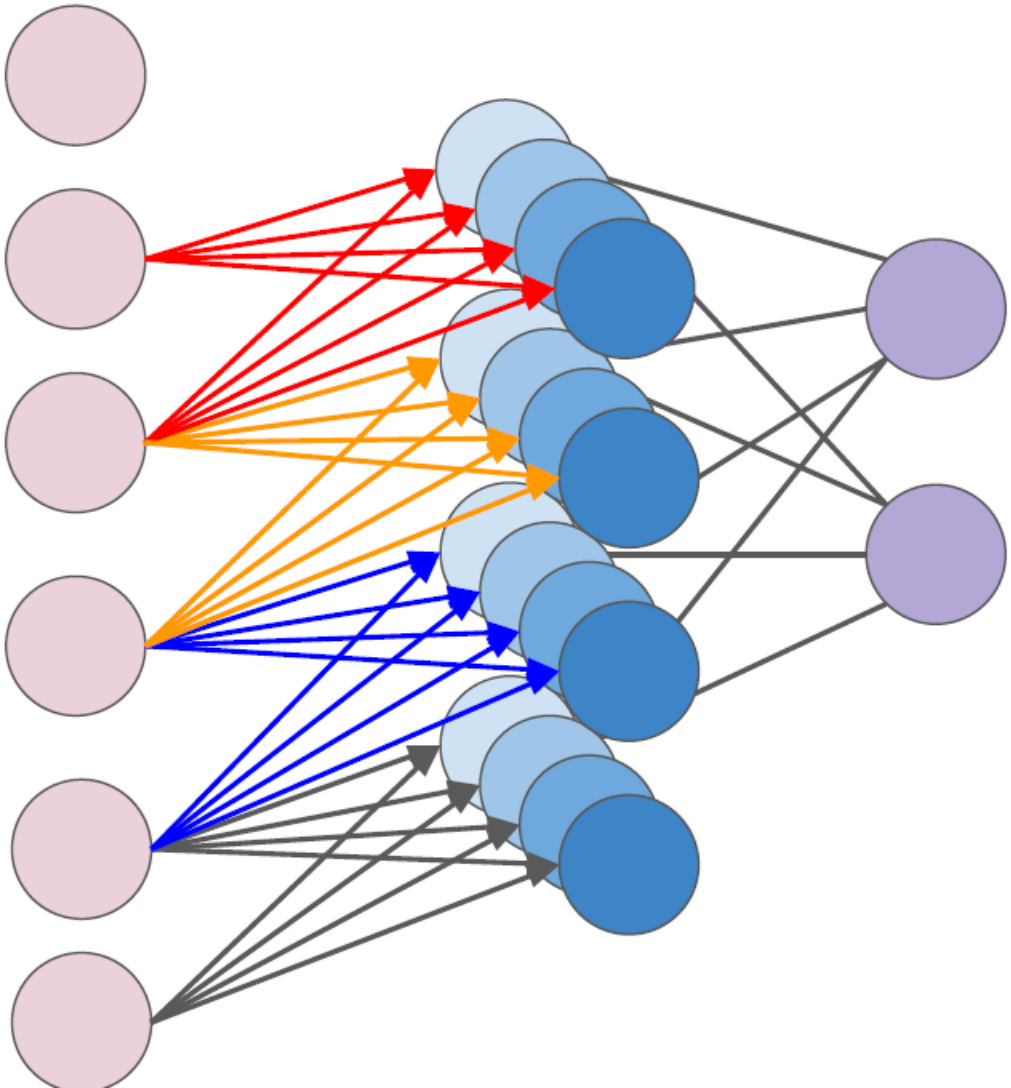
Convolutions and Filters

- 1-D Convolution
  - Filters: 2
  - Filter Size: 2
  - Stride: 1



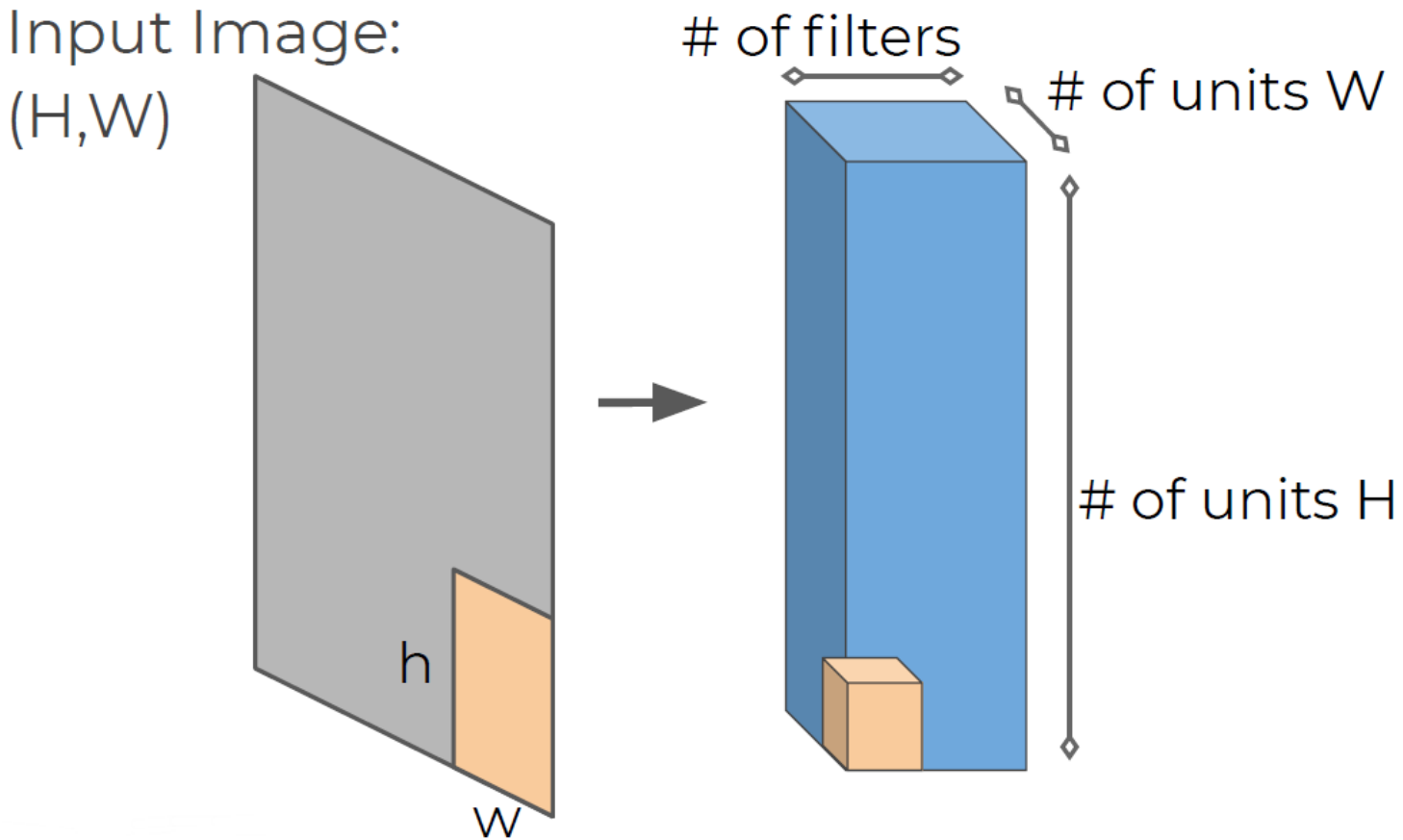
Convolutions and Filters

- 1-D Convolution
  - Filters: 4
  - Filter Size: 2
  - Stride: 1
- Each filter is detecting a different feature



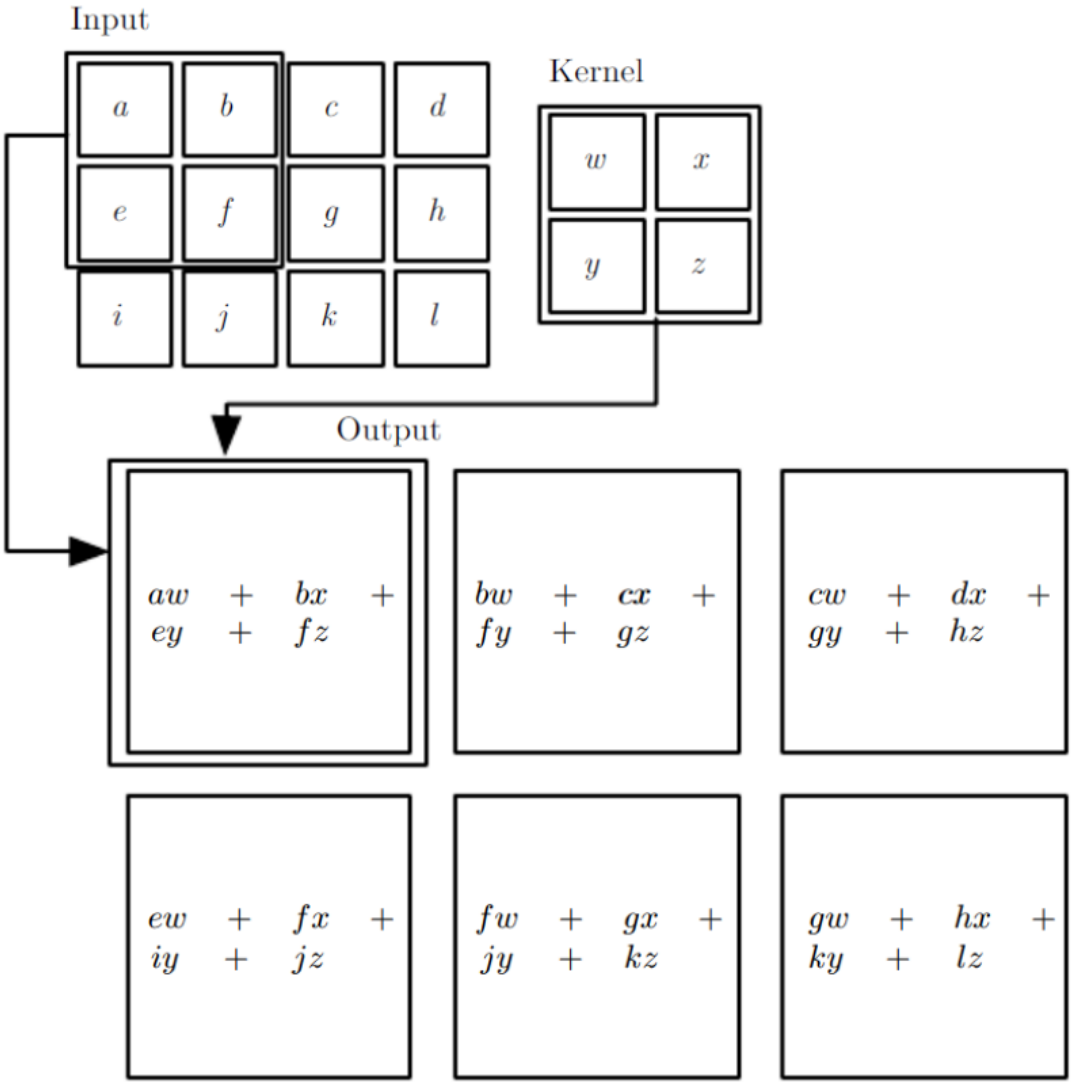
Convolutions and filters

- Let's now expand these concepts to 2-D Convolution, since we'll mainly be dealing with images.

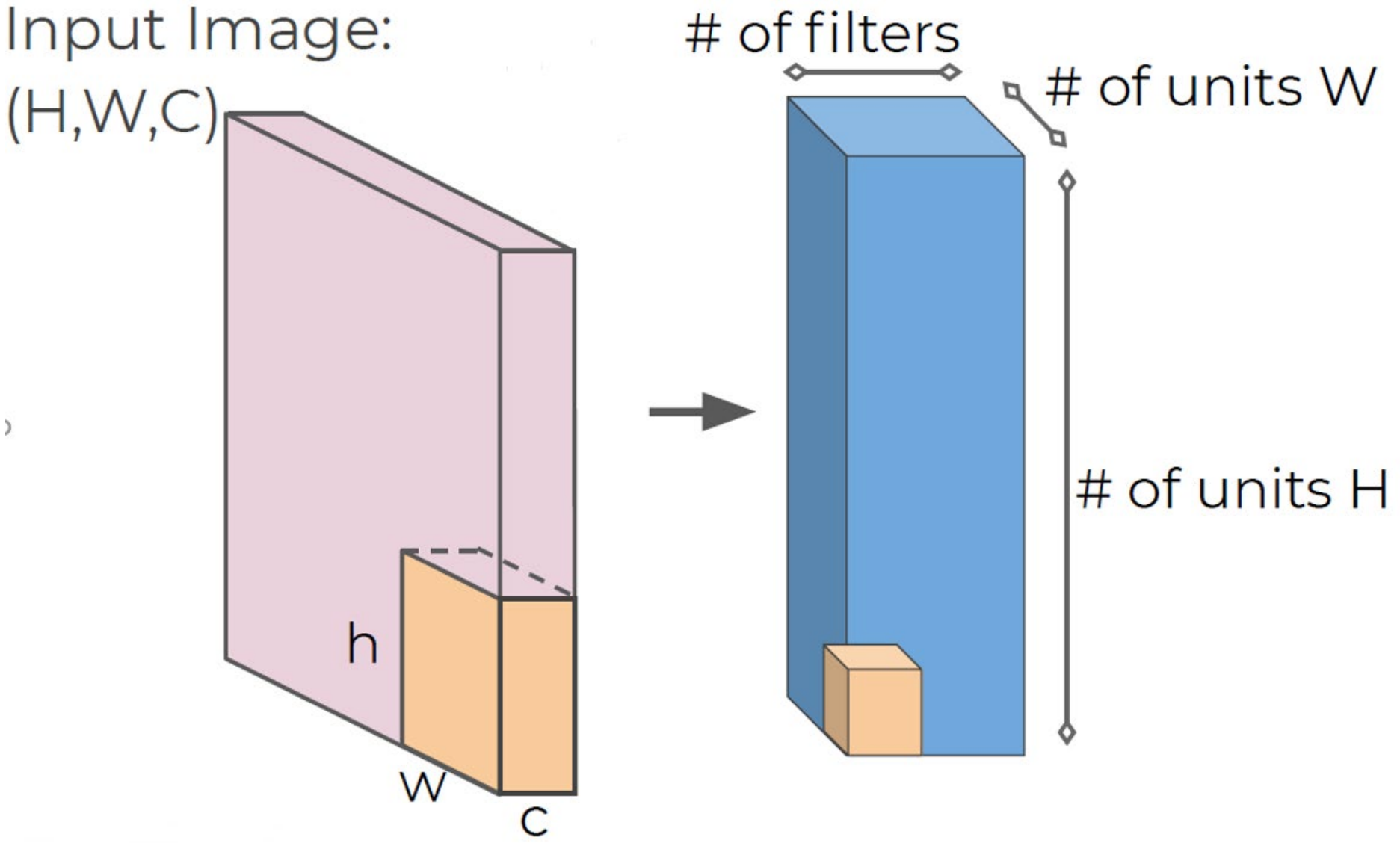


Convolutions and Filters

- The convolution operation



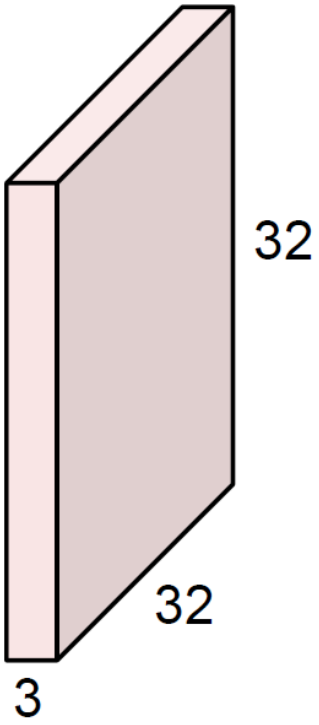
Convolutions and Filters



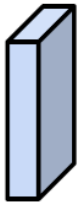
Convolutions and Filters

Filters always extend the full depth of the input volume

32x32x3 image

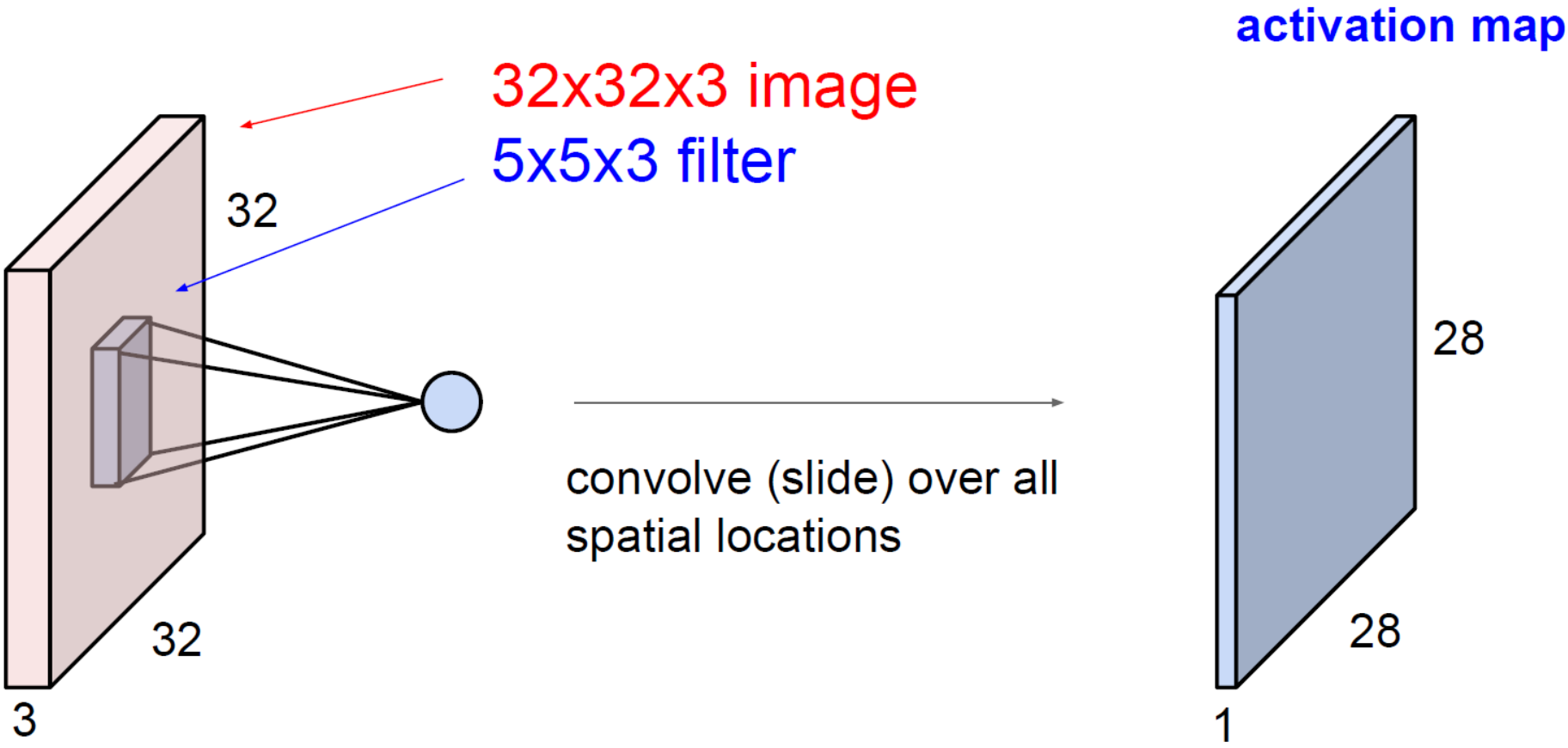


5x5x3 filter



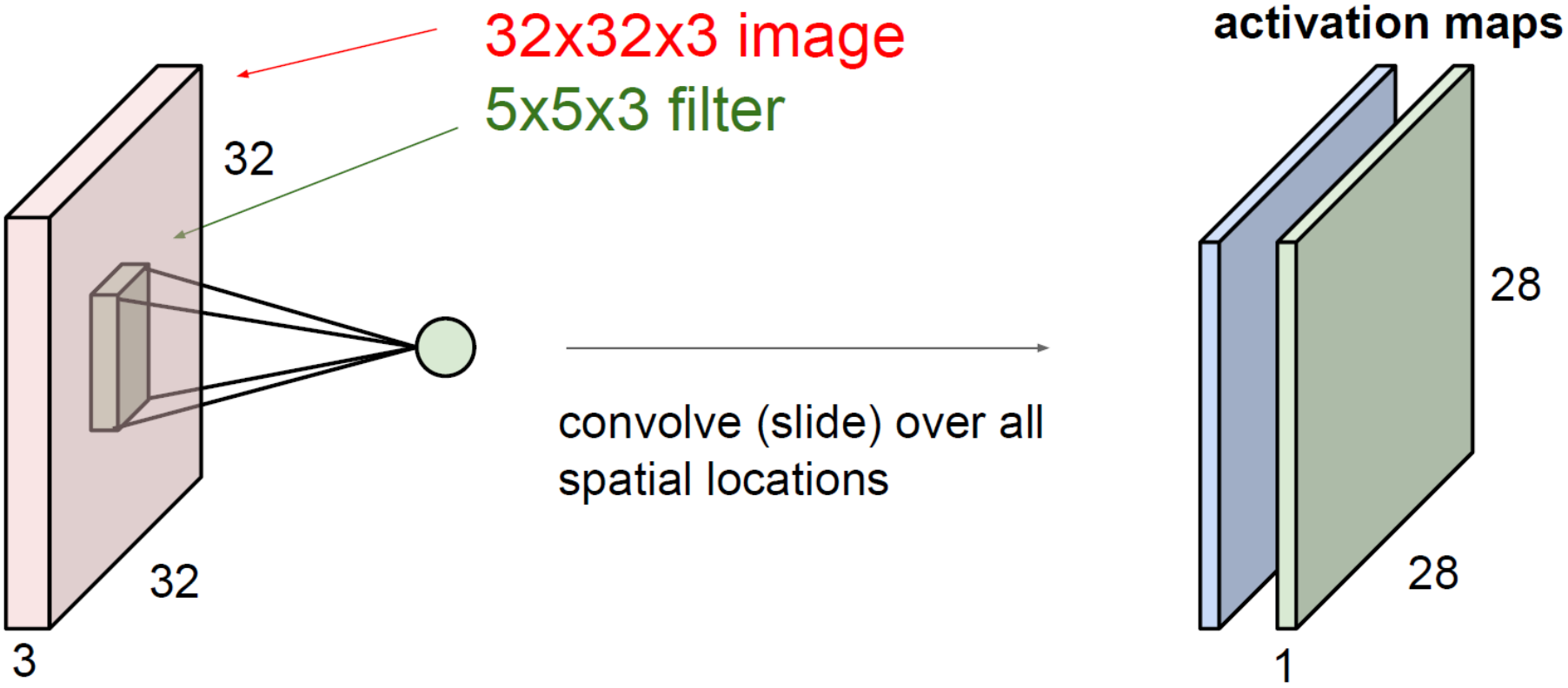
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolutions and Filters

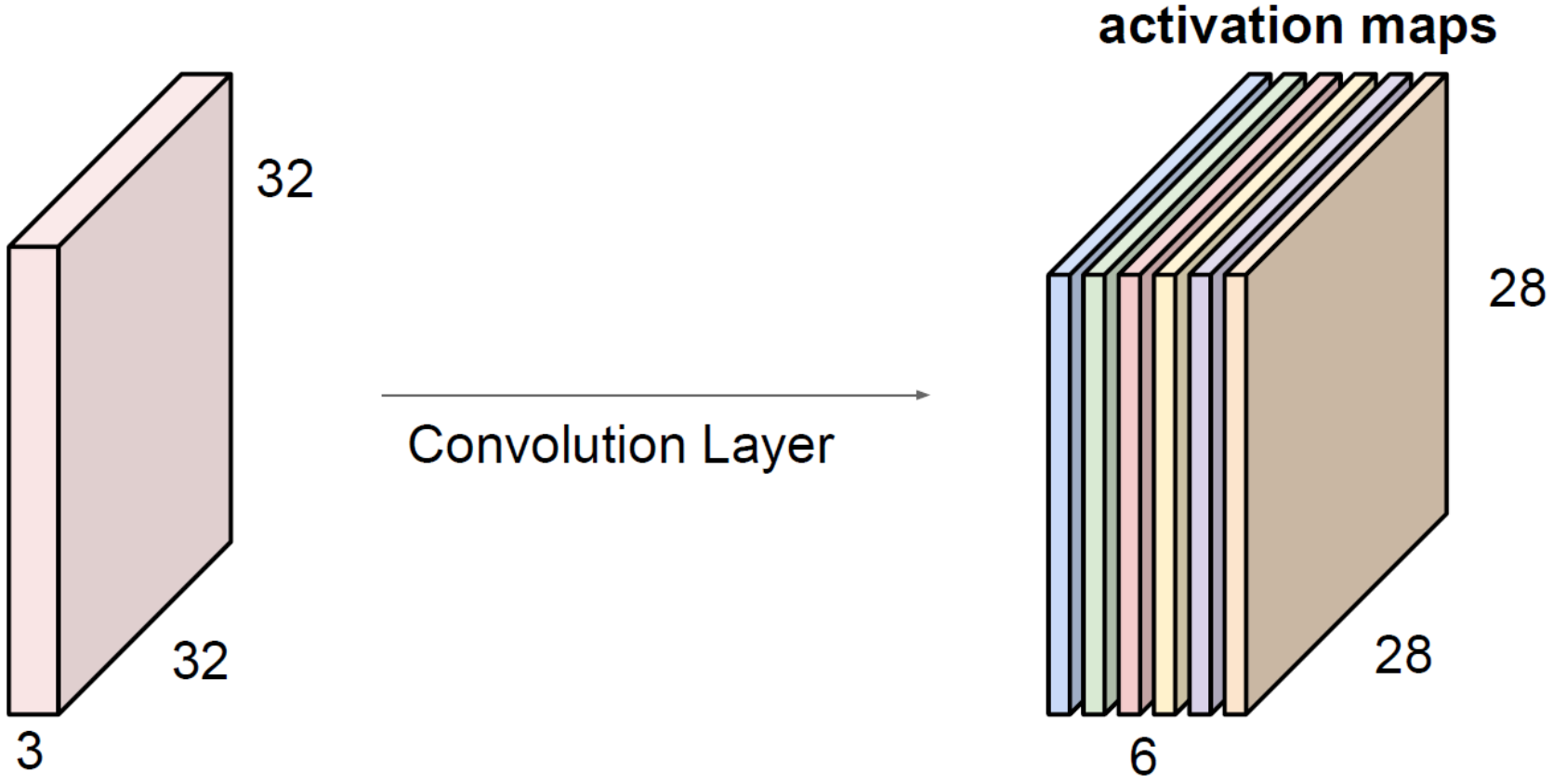


Convolutions and Filters

consider a second, green filter

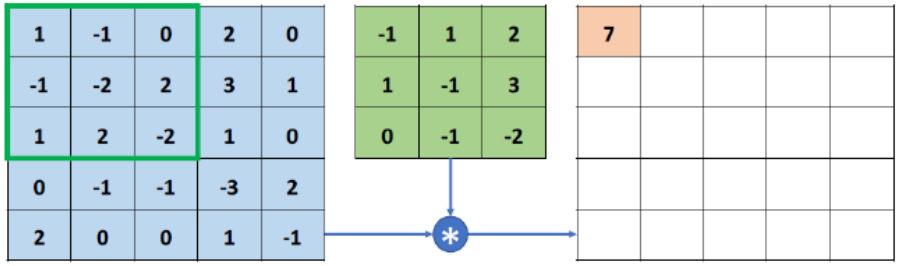


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

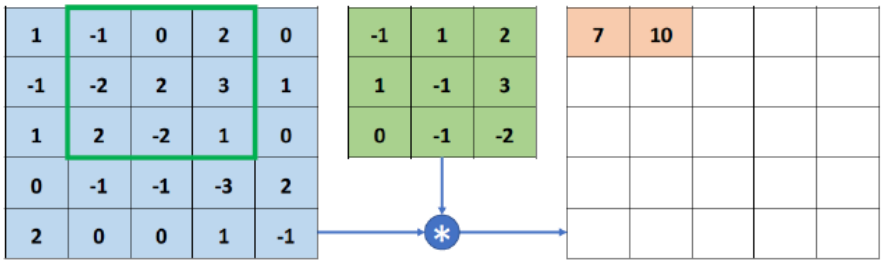


We stack these up to get a "new image" of size 28x28x6!

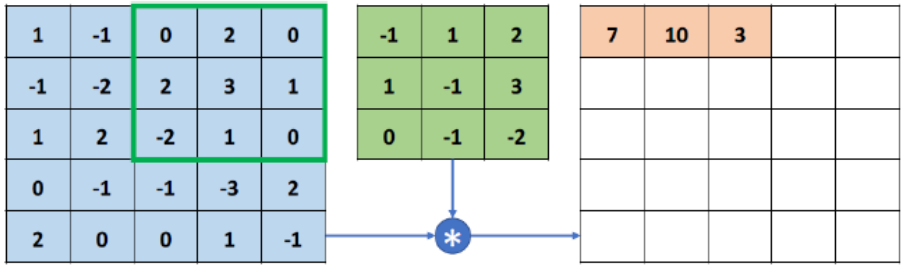
CNN: single channel, single filter



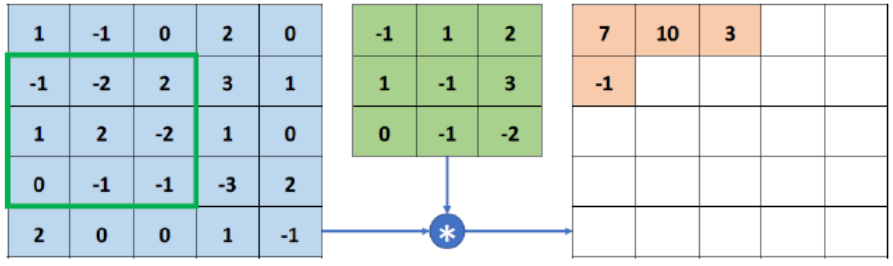
step 1



step 2

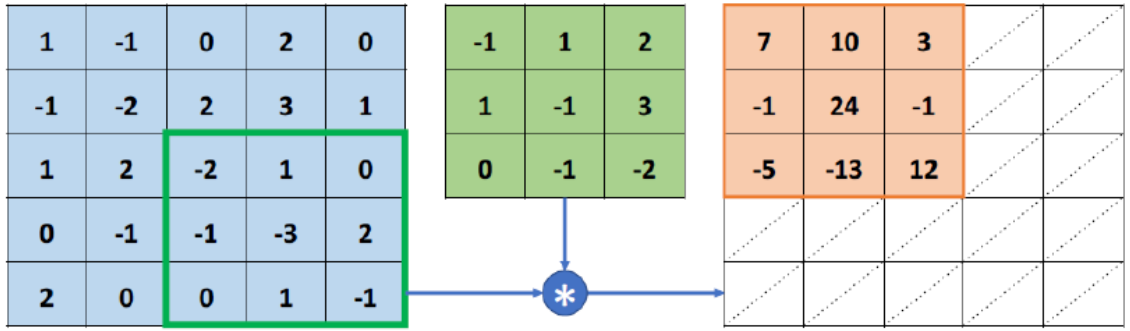


step 3

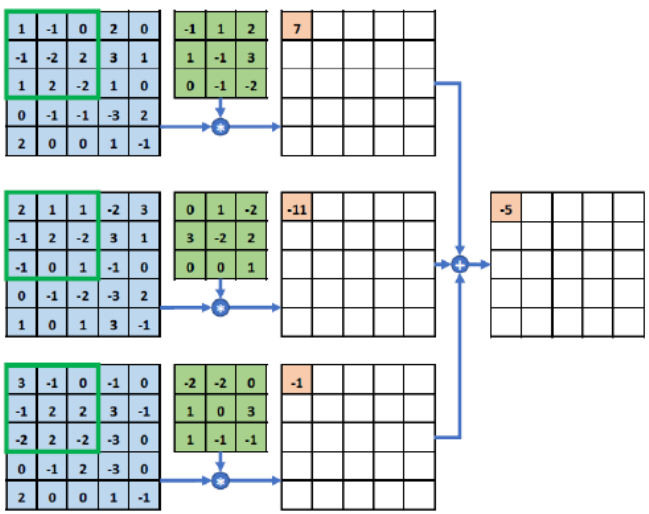


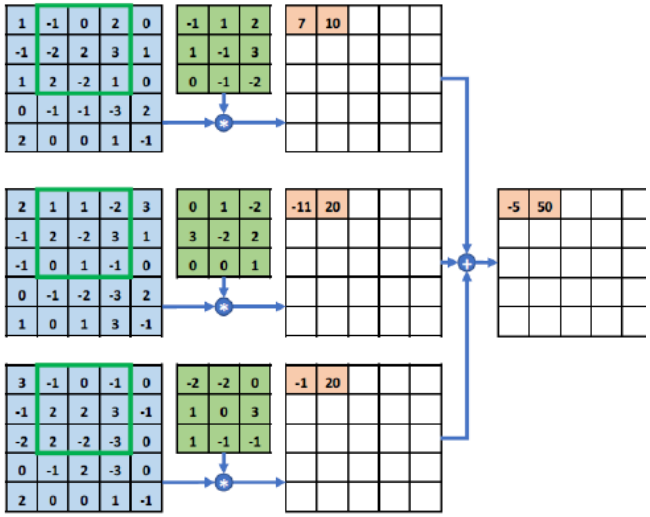
step 4

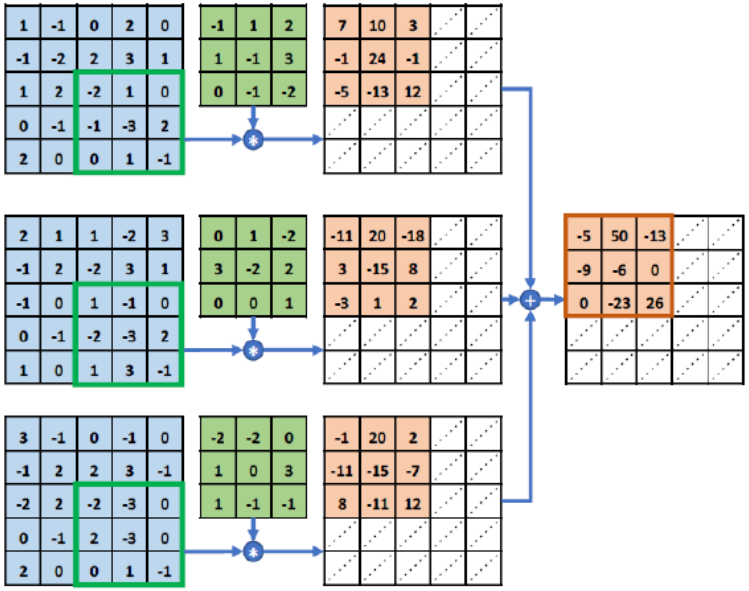
.....

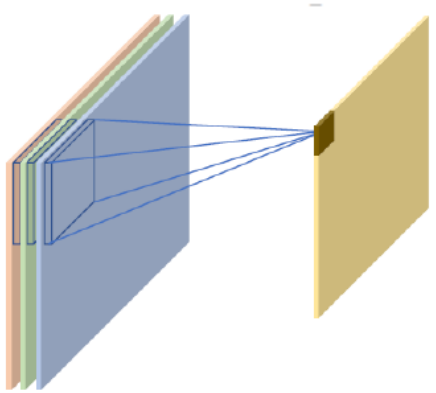


CNN: multiple channels, multiple filters



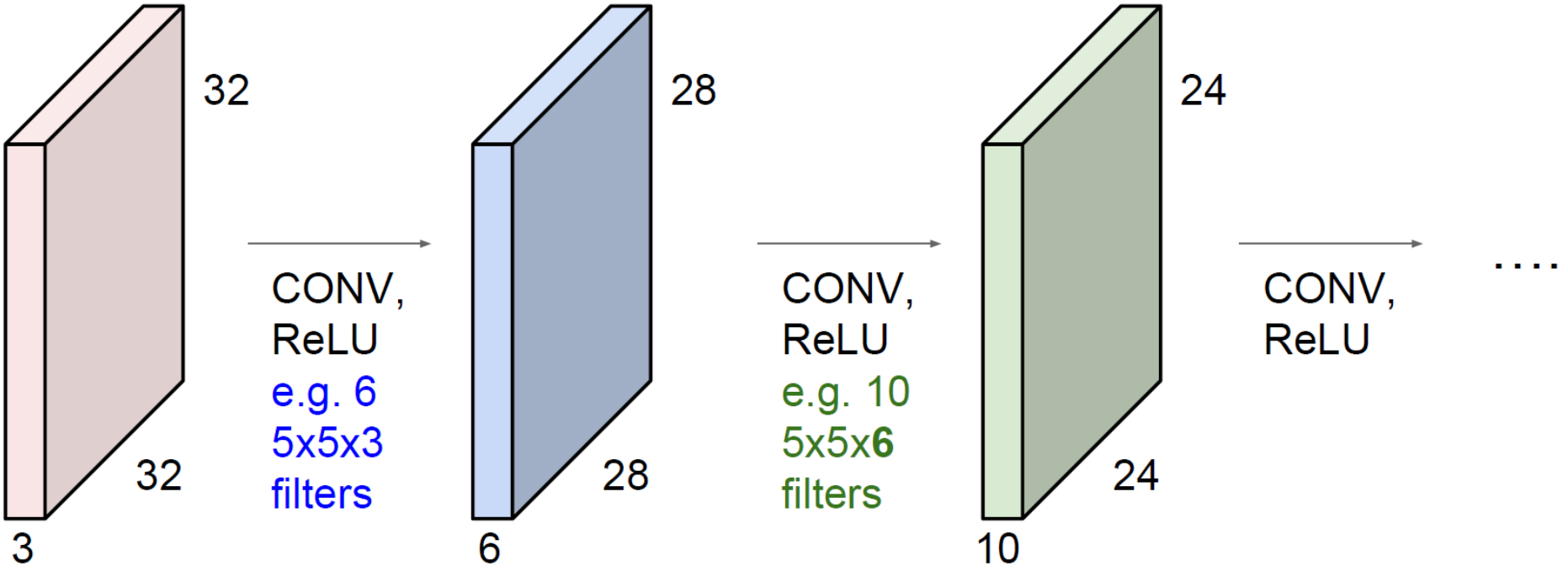




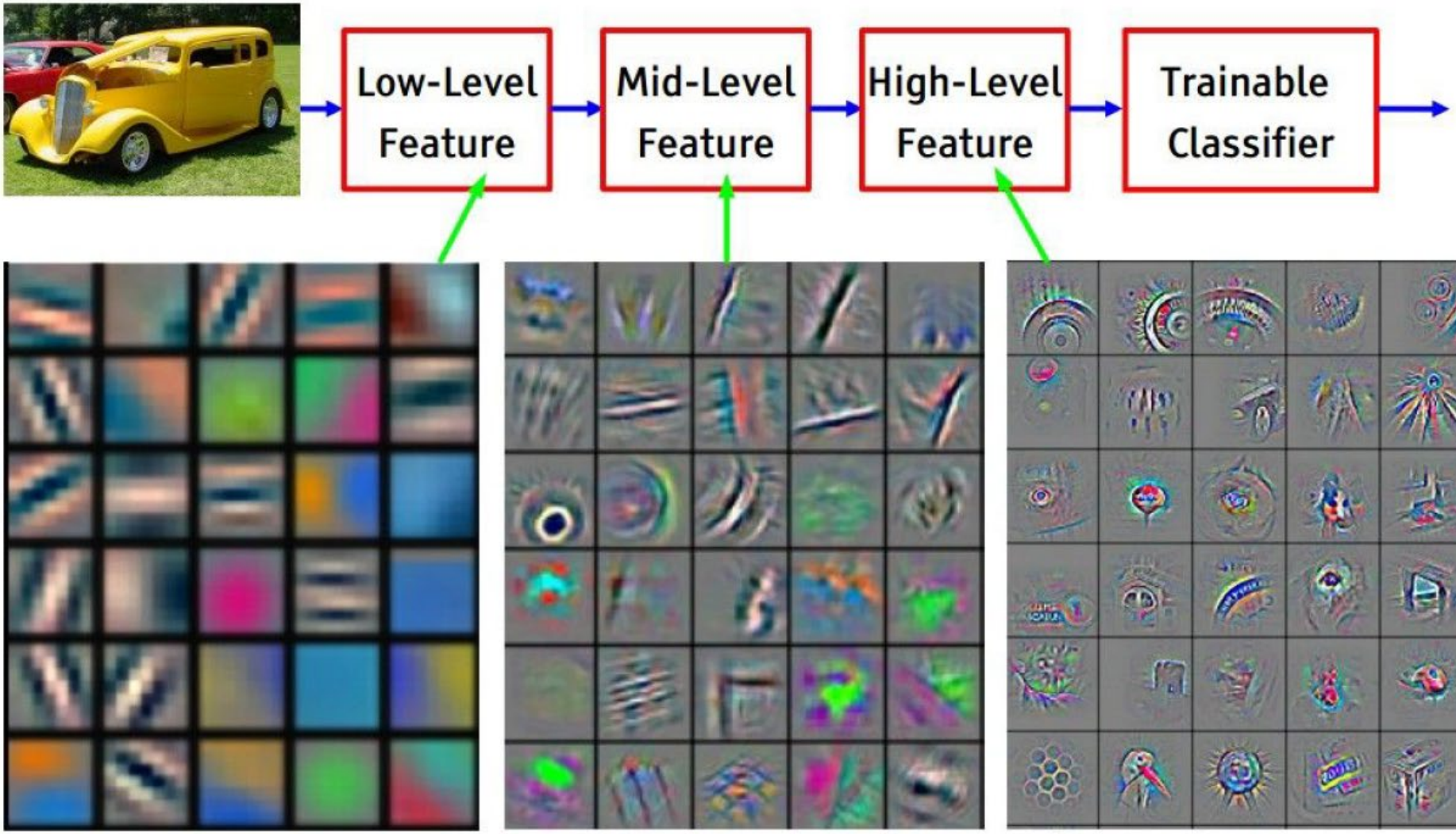


Convolutions and Filters

- A sequence of convolutional layers, interspersed with activation functions.

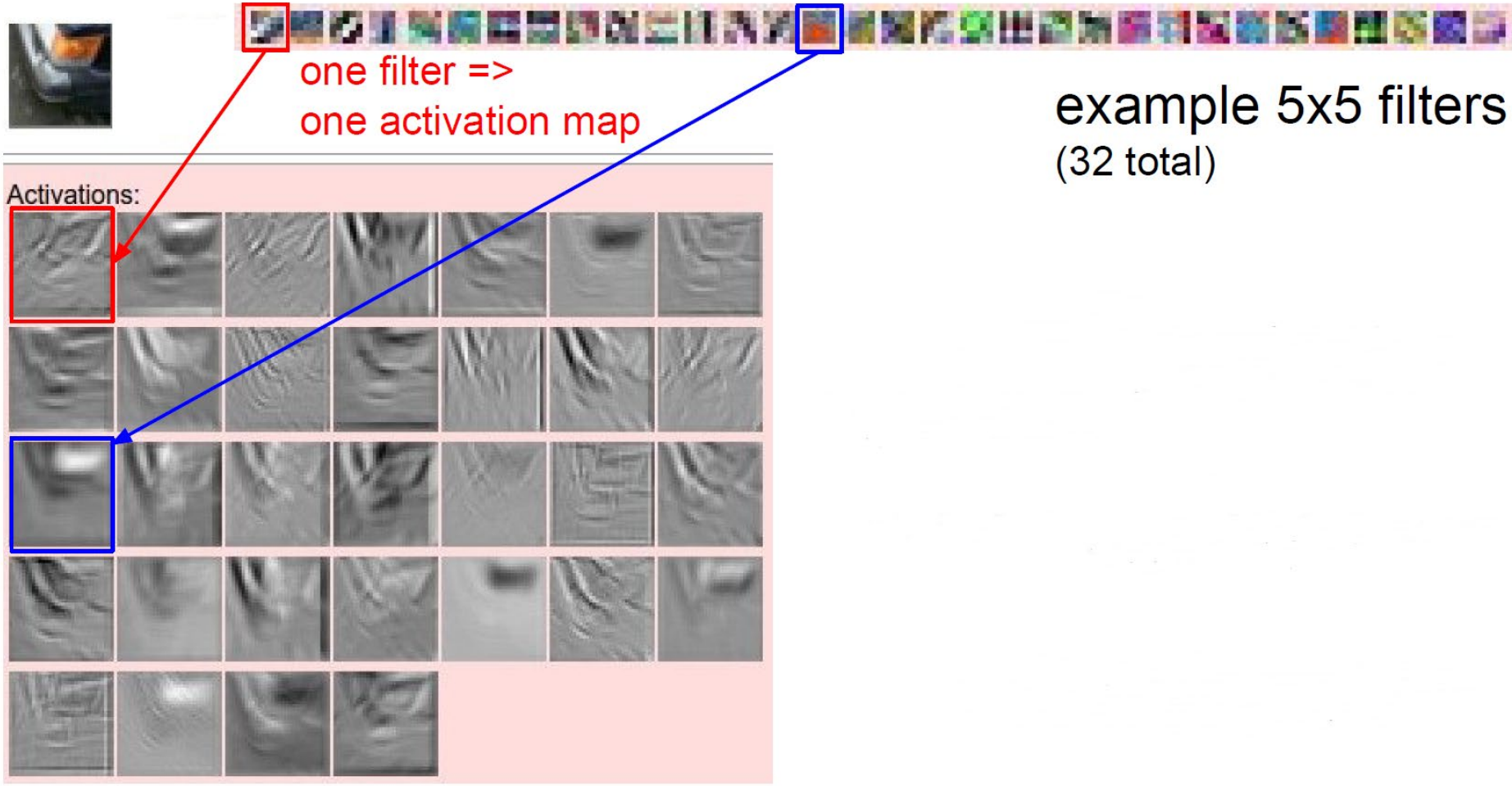


Convolutional Neural Networks



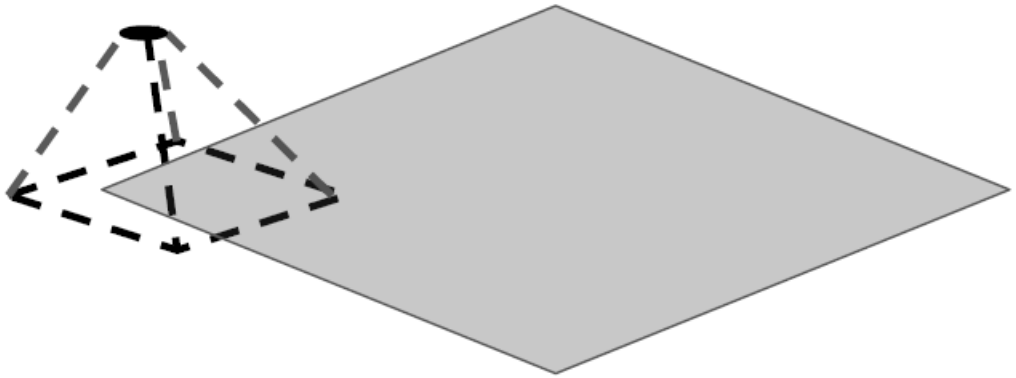
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutions and Filters



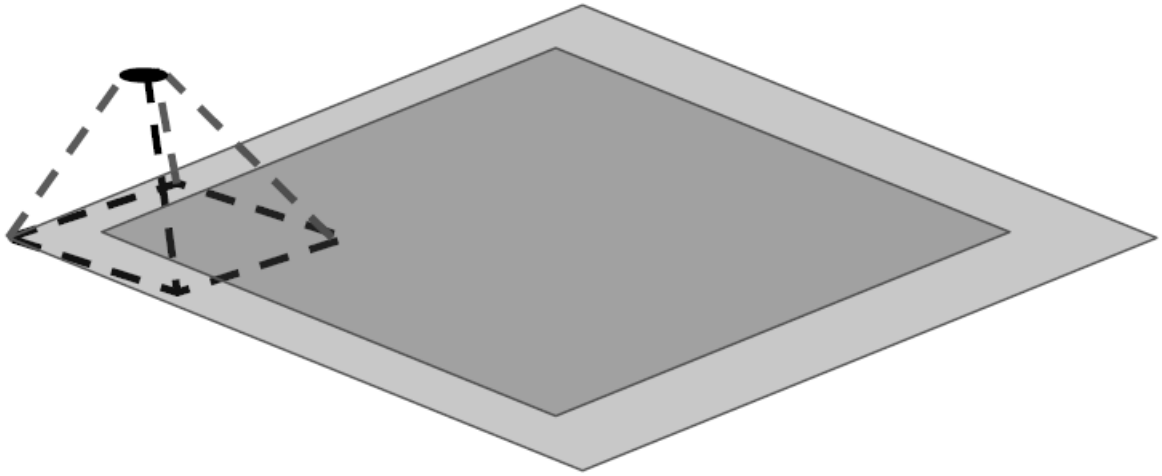
Padding

- We run into a possible issue for edge neurons! There may not be an input there for them.



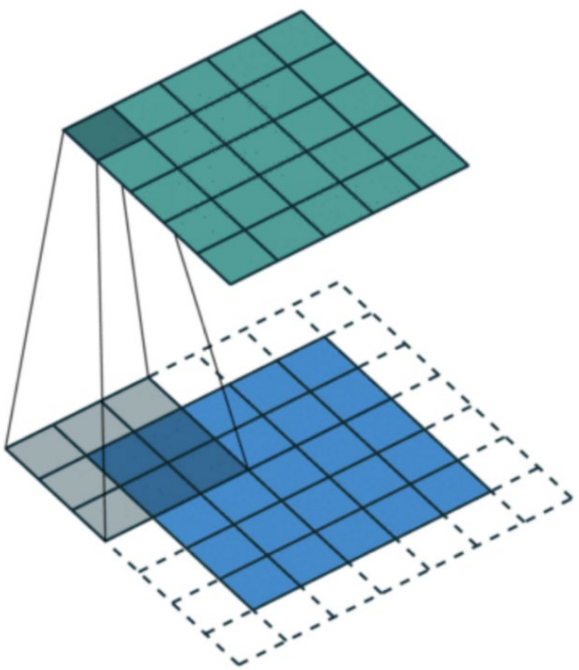
Padding

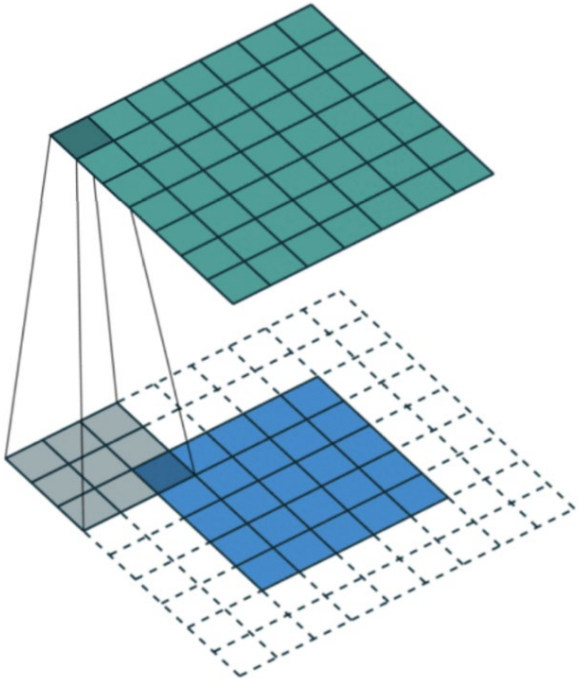
- We can fix this by adding a “padding” of zeros around the image.



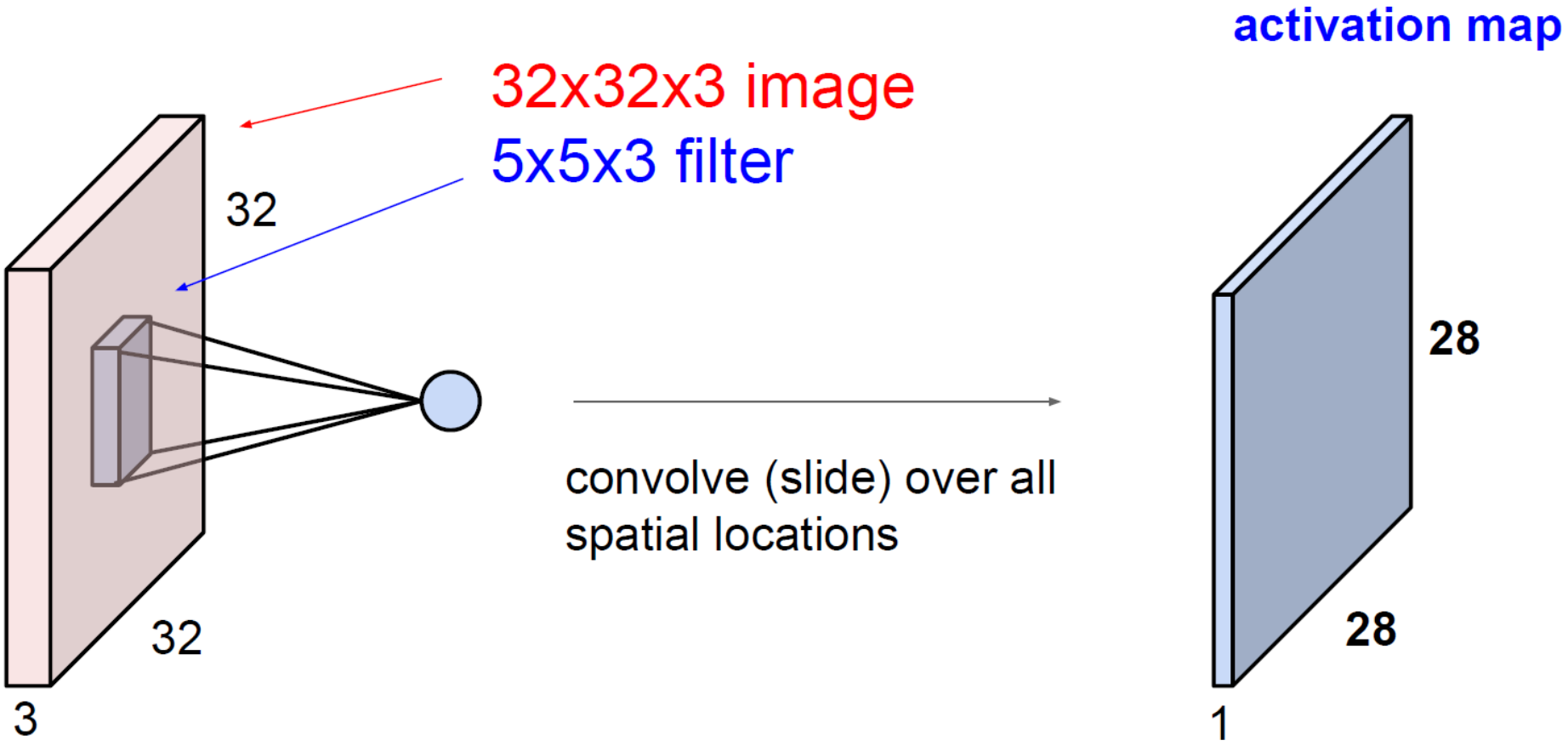
Padding

- **(Zero-)Padding** refers to the process of symmetrically adding zeroes to the input matrix. It's a commonly used modification that allows the size of the input to be adjusted to our requirement.

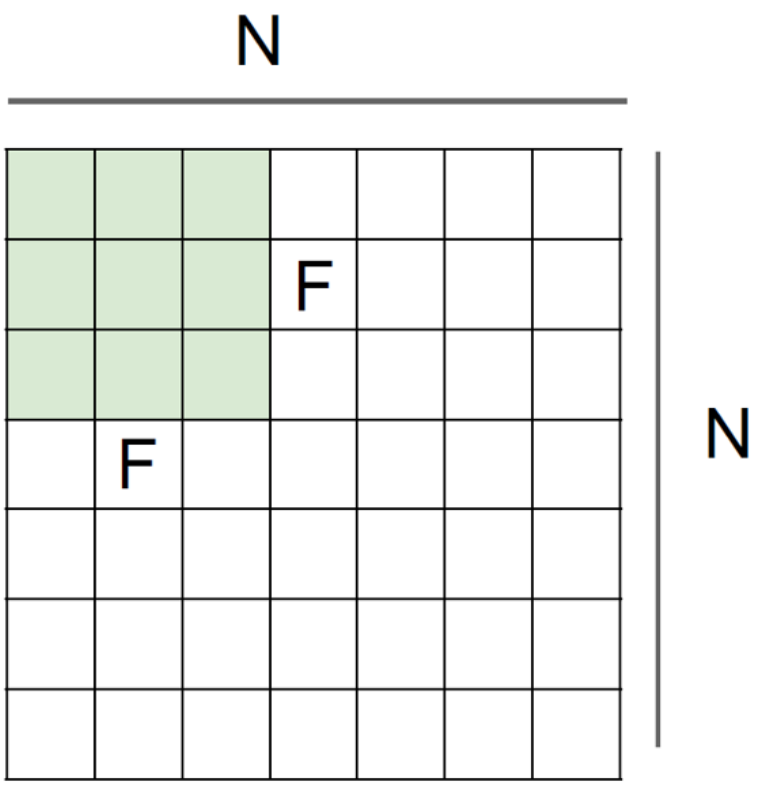




A closer look at spatial dimensions:



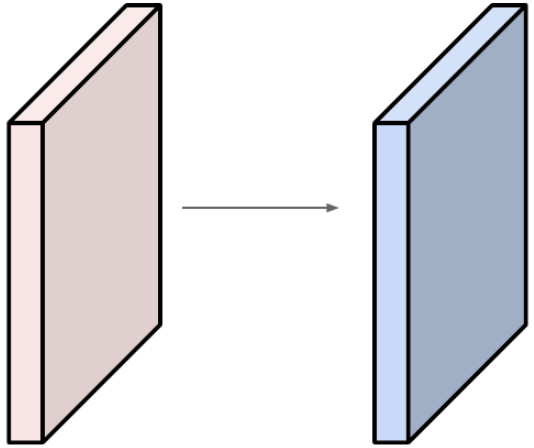
A closer look at spatial dimensions:



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

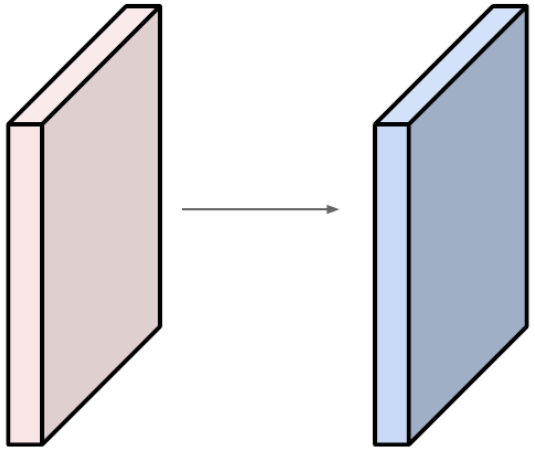
Examples time:



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

Output volume size: ?

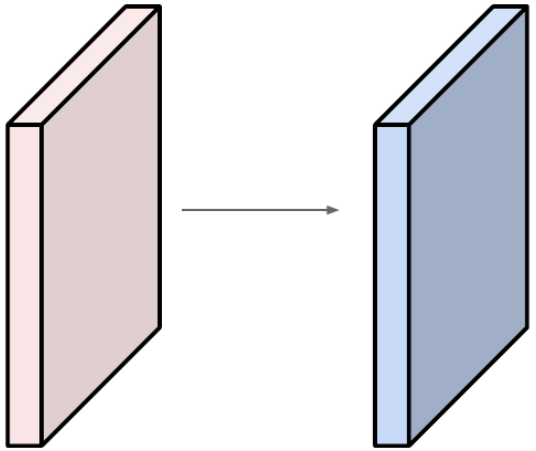
Examples time:



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

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

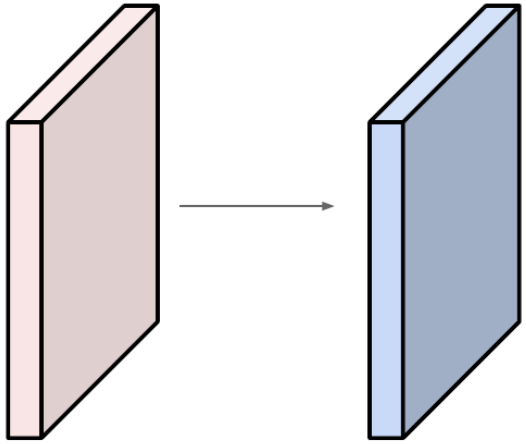
Examples time:



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**

Convolutional neural networks

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.

Convolutional neural networks

Common settings:

K = (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 = ? (whatever fits)
- F = 1, S = 1, P = 0

Summary. To summarize, the Conv Layer:

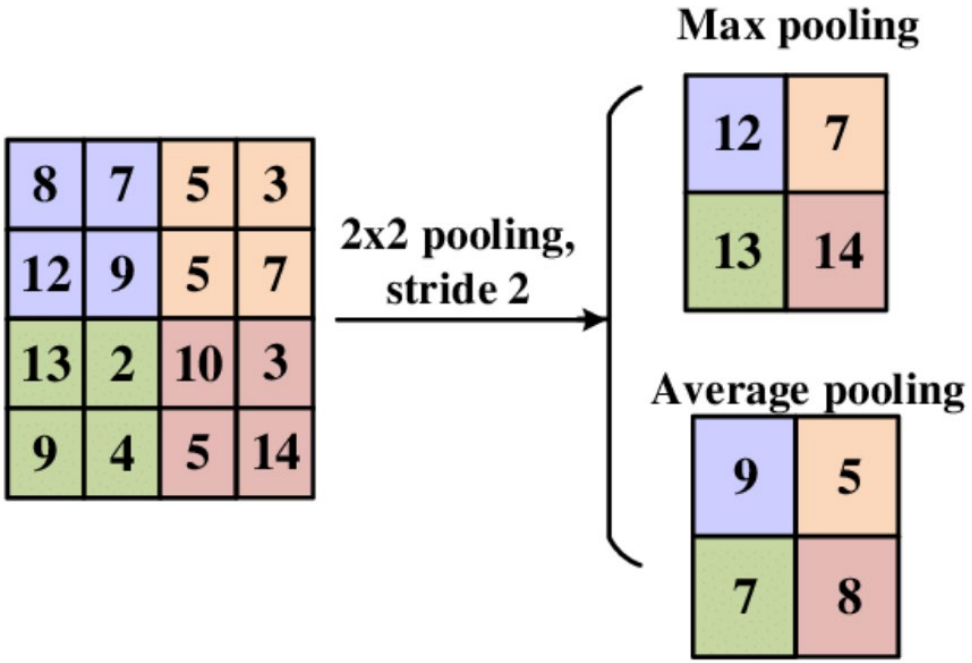
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (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

- Pooling layers will **subsample** the input image, which reduces the memory use and computer load as well as reducing the number of parameters.
  - Subsampling the pixels will not change the object

Pooling

- Max pooling: The maximum pixel value of the batch is selected.
- Average pooling: The average value of all the pixels in the batch is selected.



Pooling

- In the following example, a filter of 9x9 is chosen.

- Max pooling selects the brighter pixels from the image.
- Average pooling method smooths out the image and hence the sharp features may not be identified.

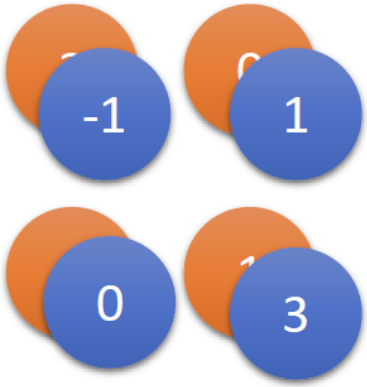
Pooling

- 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

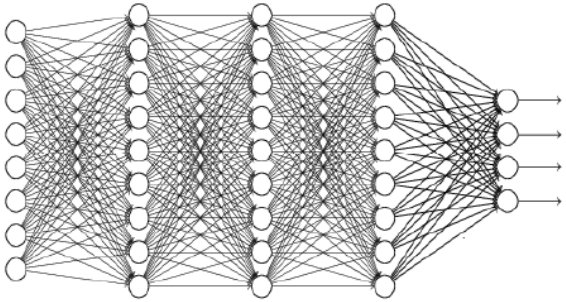
Common settings:

F = 2, S = 2
F = 3, S = 2

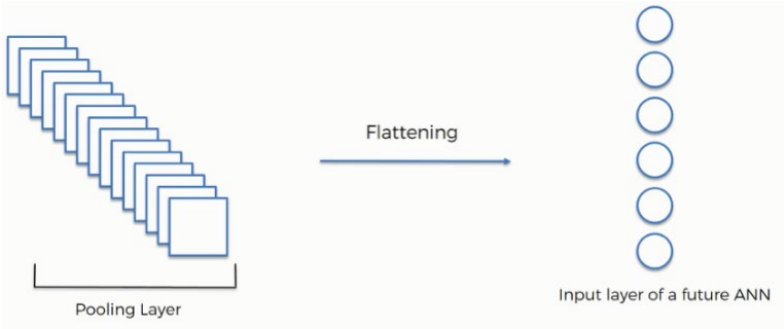
Flatten



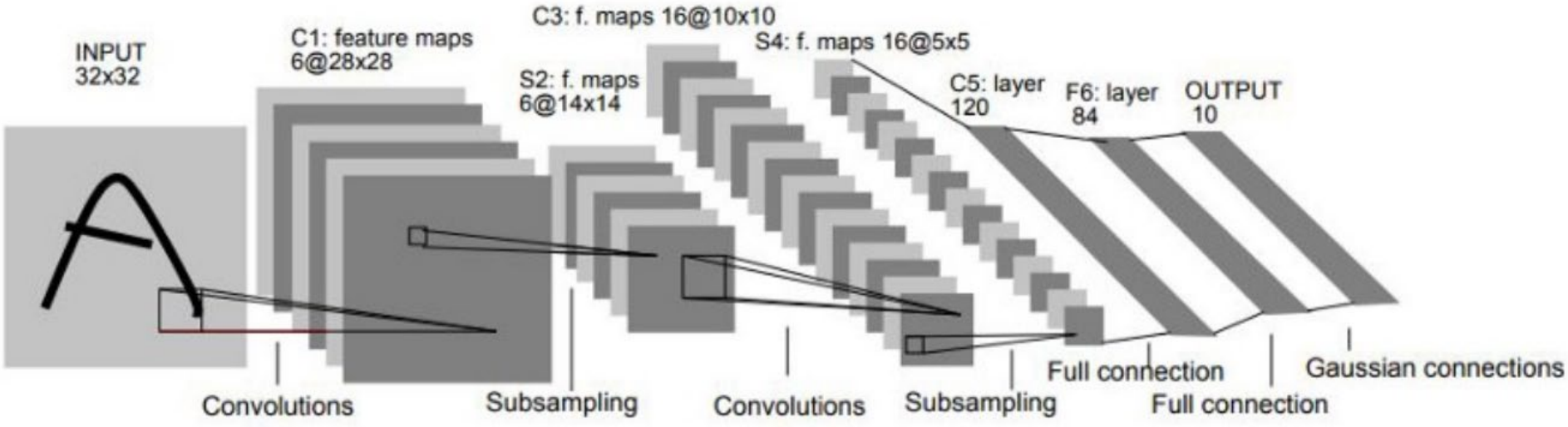
Flatten



Fully Connected Feedforward network



Case Study: LeNet-5 [LeCun et al., 1998]



LeNet-5 [LeCun et al., 1998]

- Summary of LeNet-5 Architecture

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax