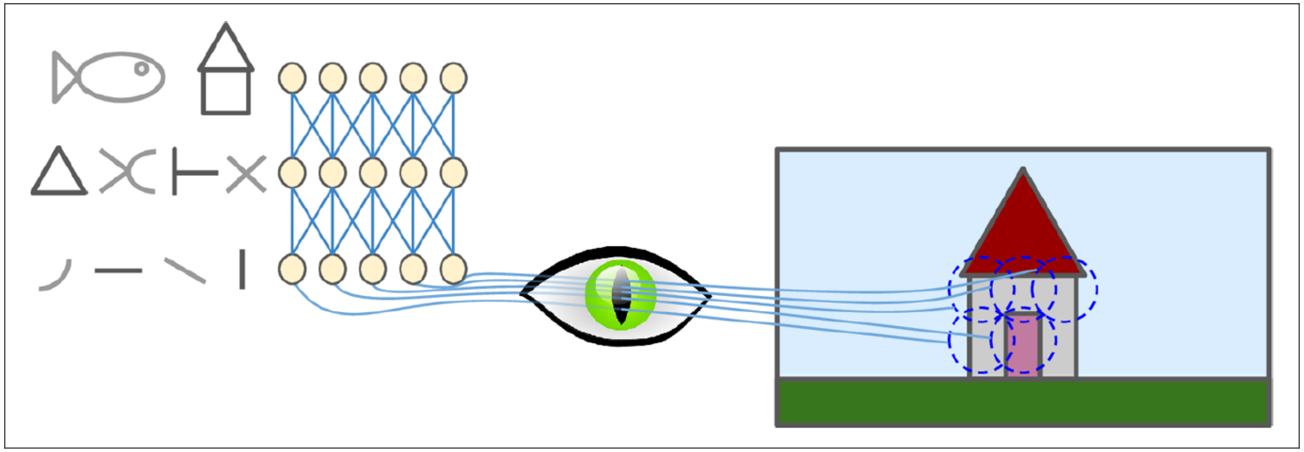
Deep Learning-Convolutional Neural Network

COEN140

Santa Clara University

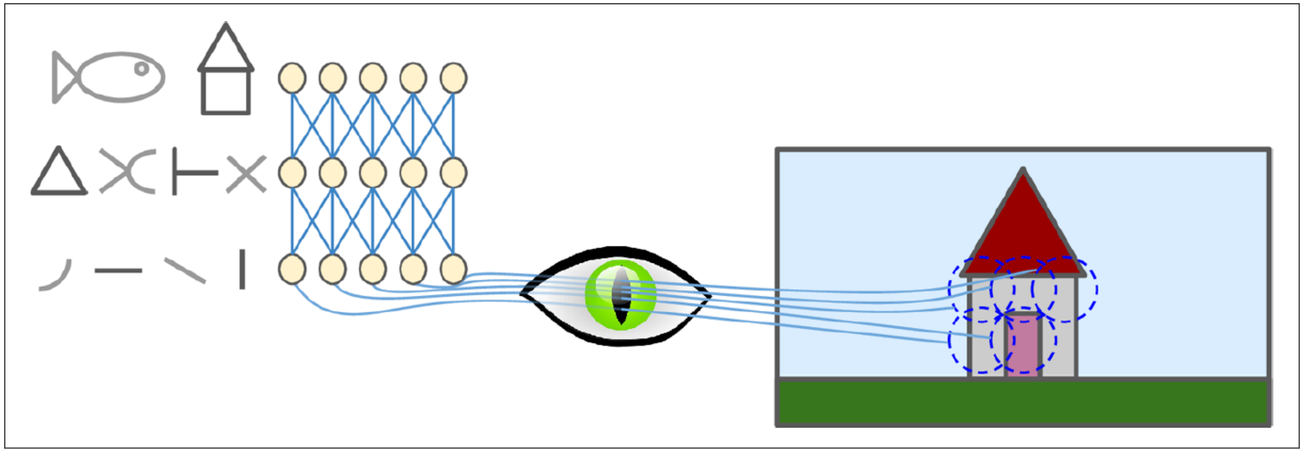
Motivation

- Convolutional neural network (CNN): a computational model that simulates how human eyes see the world
- Nodes: neurons in human visual cortex
- Neurons are connected
  - But not fully connected



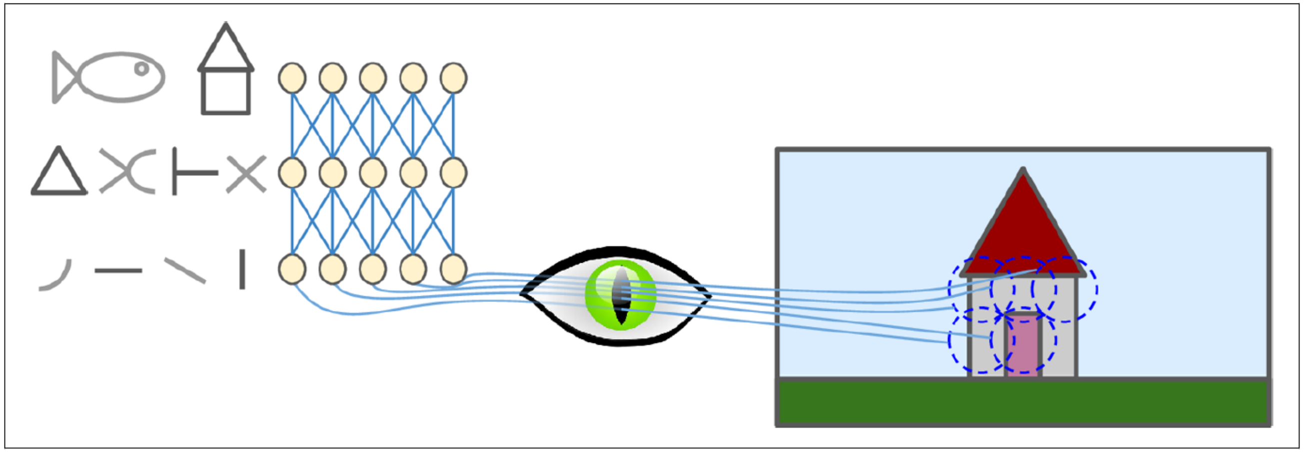
Motivation

- Neurons have a small local receptive field
  - They react only to visual stimuli in a limited region of the visual field



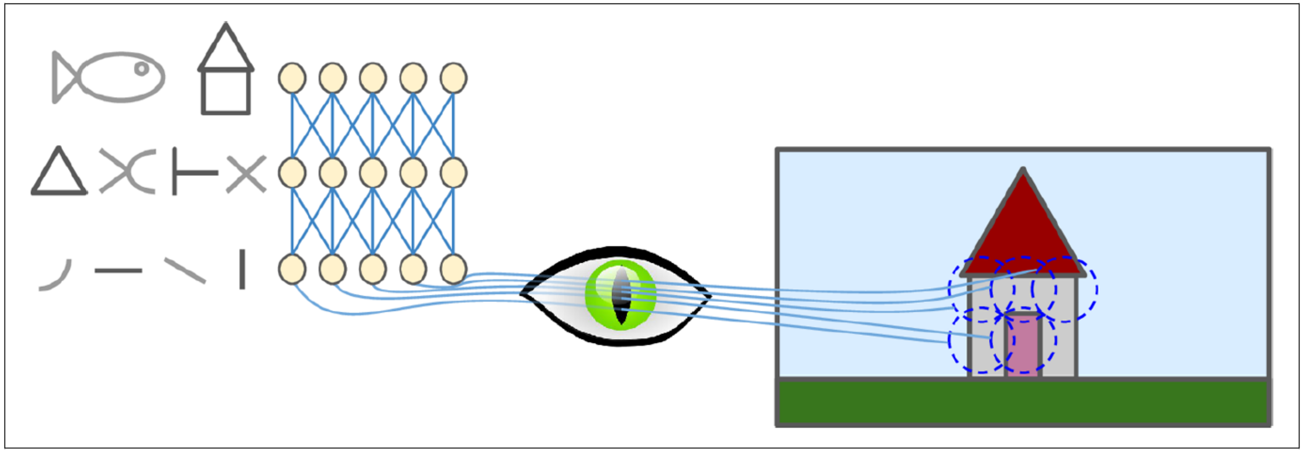
Motivation

- Some neurons react only to horizontal lines
- Some neurons react to vertical lines
- Some neurons react to curves
- …



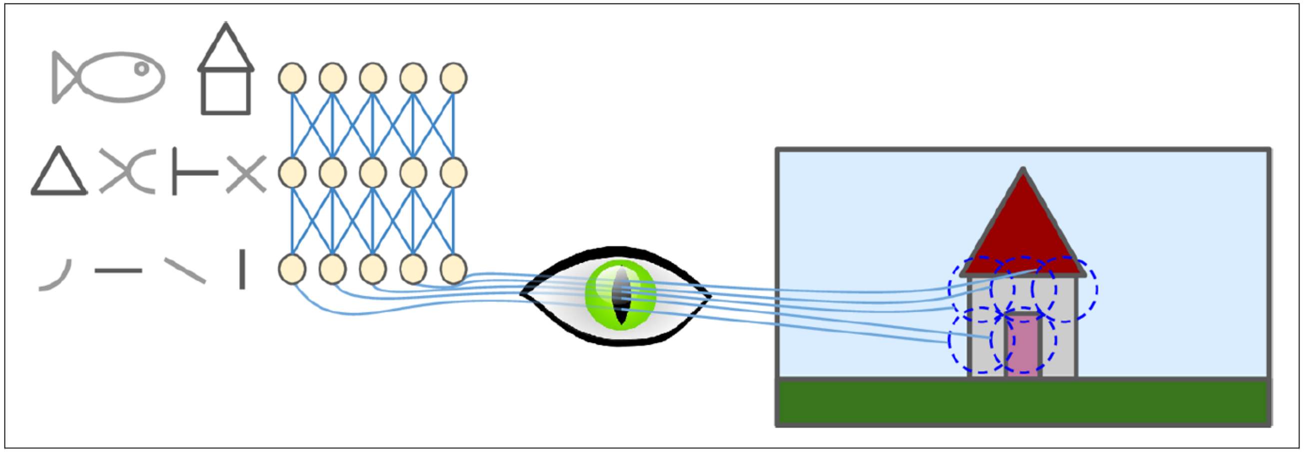
Motivation

- Some neurons react to more complex patterns that are combinations of the lower-level patterns
  - Higher-level neurons are based on the outputs of neighboring lower-level neurons



Motivation

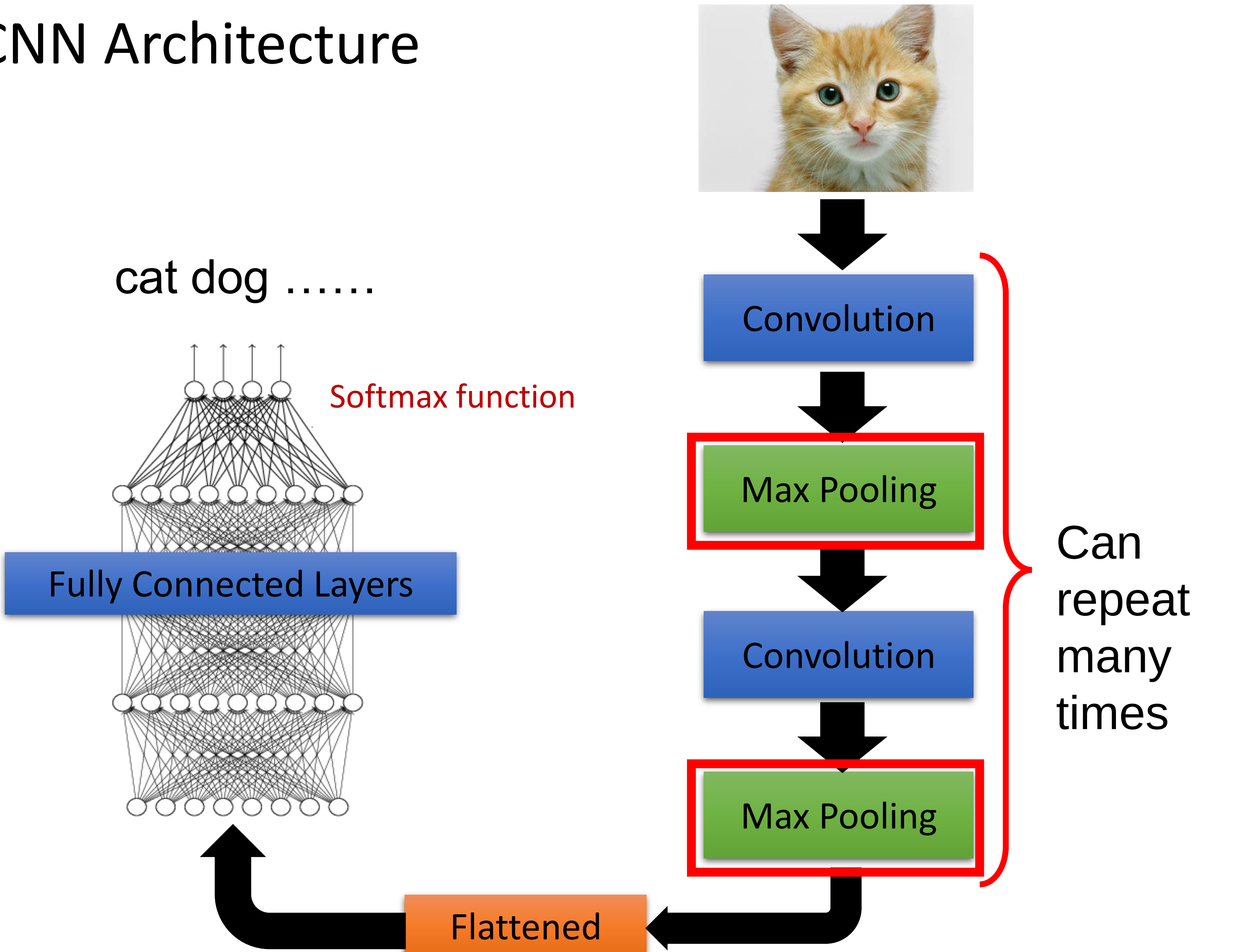
- This powerful architecture can detect all sorts of complex patterns in any area of the visual field
- The above is the motivation of CNN



Convolutional Neural Network (CNN)

- How exactly does a CNN work?
  - In terms of computation…
- A CNN has some convolutional layers (and some other layers).
- A convolutional layer has a number of filters that execute convolution operations.

CNN Architecture

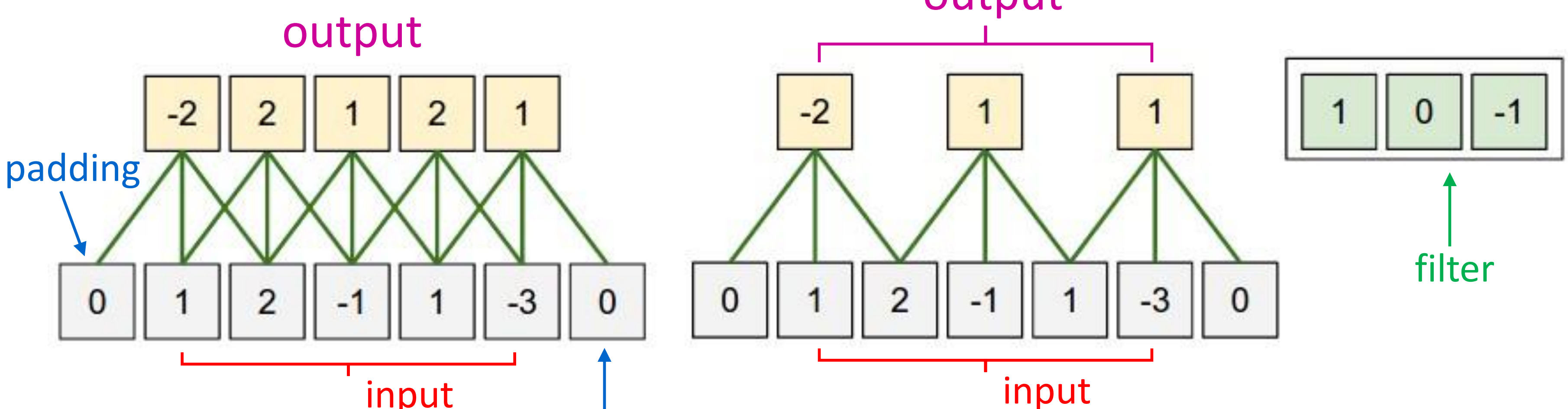


CNN Architecture

- Input: an image, some text, etc.
- Convolutional layer: generate feature maps
- Max pooling layer: capture the most important features – those with the highest values
- Softmax function: outputs the probability distribution over different classes (labels)

Convolution-1D

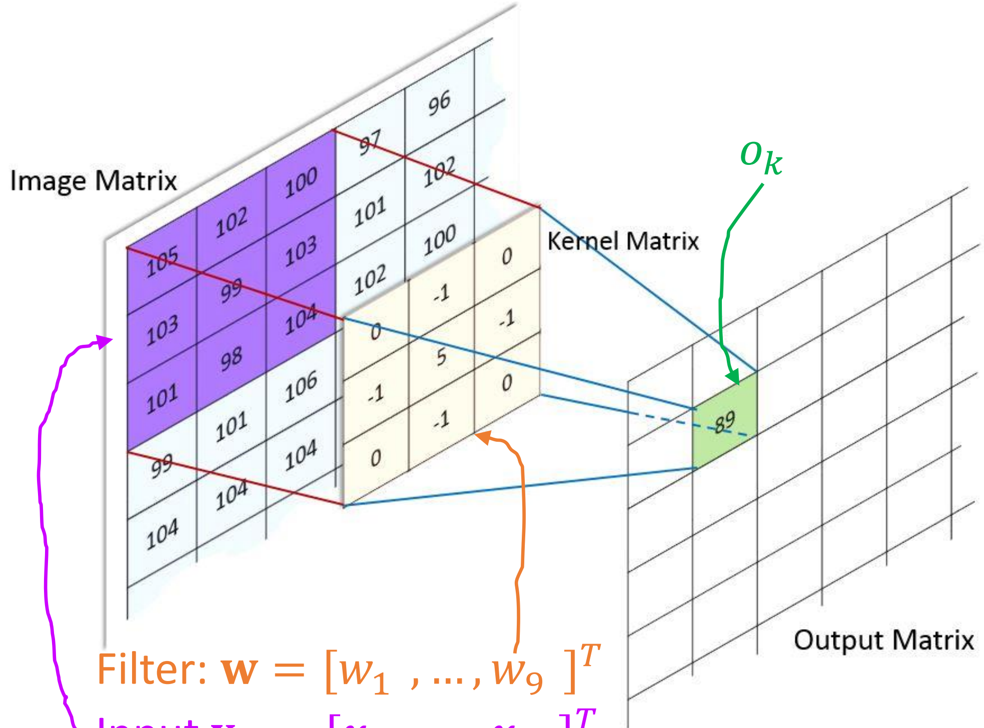
- Stride (S): how the filter is shifted
- Zero padding (P): pad the input volume with zeroes around the boarder



- Input size: W=5; Zero-padding: P=1
- Filter size: F=3
- Stride: S=1 (Left); S=2(right)
- Output size: floor((W+2P-F)/S)+1: 5(Left); 3(Right)

Convolution-2D

- Input: an image matrix
- Filter/Kernel: a smaller size matrix
  - e.g. 3×3 matrix
- The filter shifts on the input image, horizontally and vertically
- Output matrix
  - Each entry: the inner product (sum of the element-wise product) between the filter and a patch on the input image



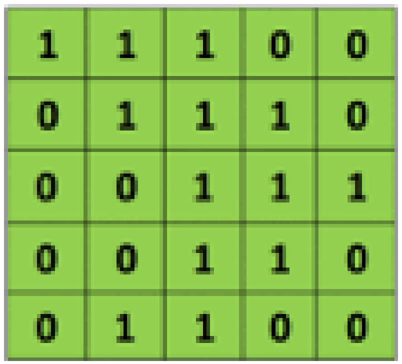
Filter: $\mathbf{w} = [w_1 \ , \dots , w_9 \ ]^T$

Input $\mathbf{x}_k = [x_{k1}, \dots , x_{k9}]^T$

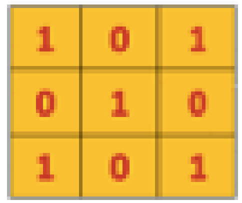
The k-th Filter output:

$$o_k = \mathbf{w}^T \mathbf{x}_k = \sum_{i=1}^{9} w_i x_{ki}$$

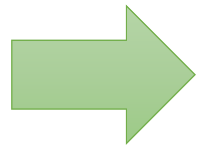
Convolution-2D

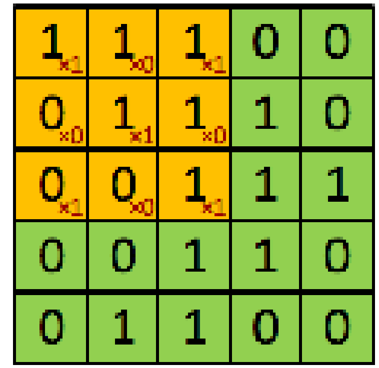


Input matrix

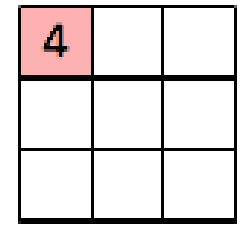


Convolutional
3×3 filter





Image

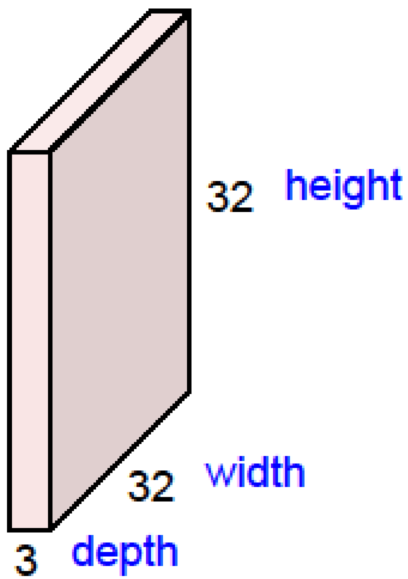


Convolved
Feature

One Conv Layer

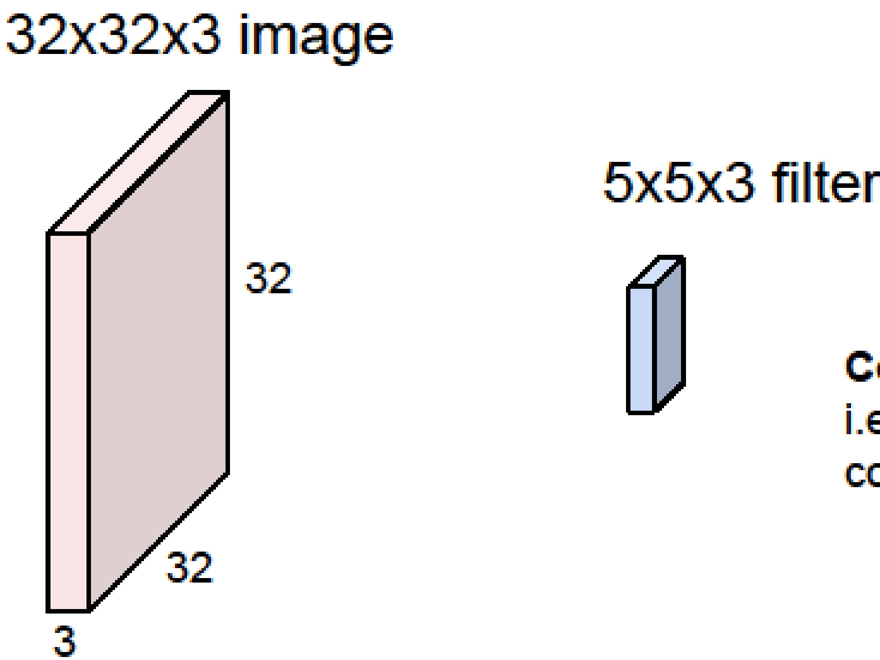
Convolution Layer

32x32x3 image -> preserve spatial structure



One Conv Layer

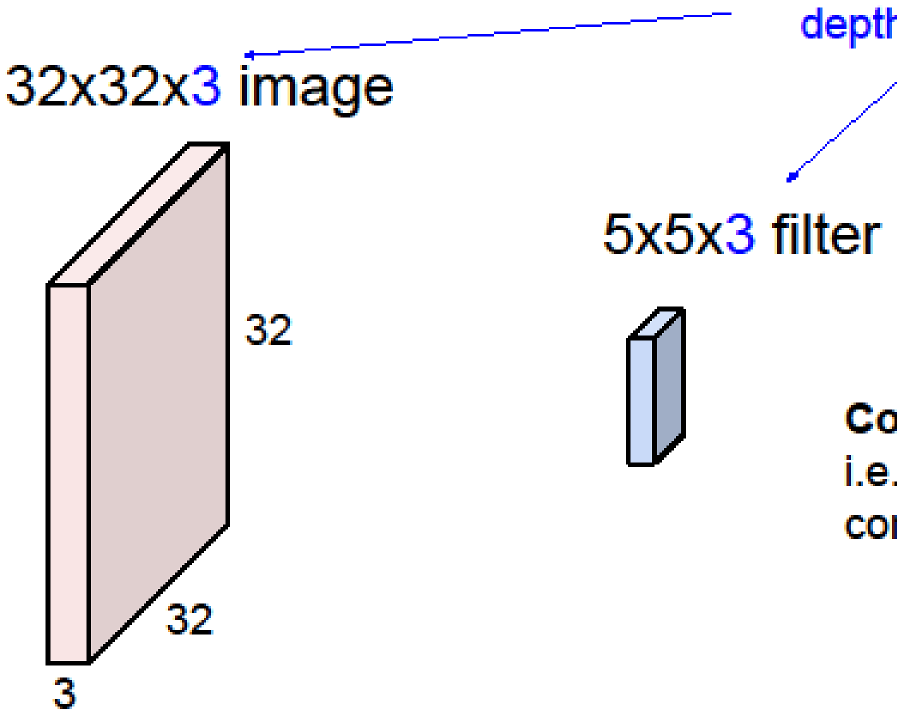
Convolution Layer



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

One Conv Layer

Convolution Layer

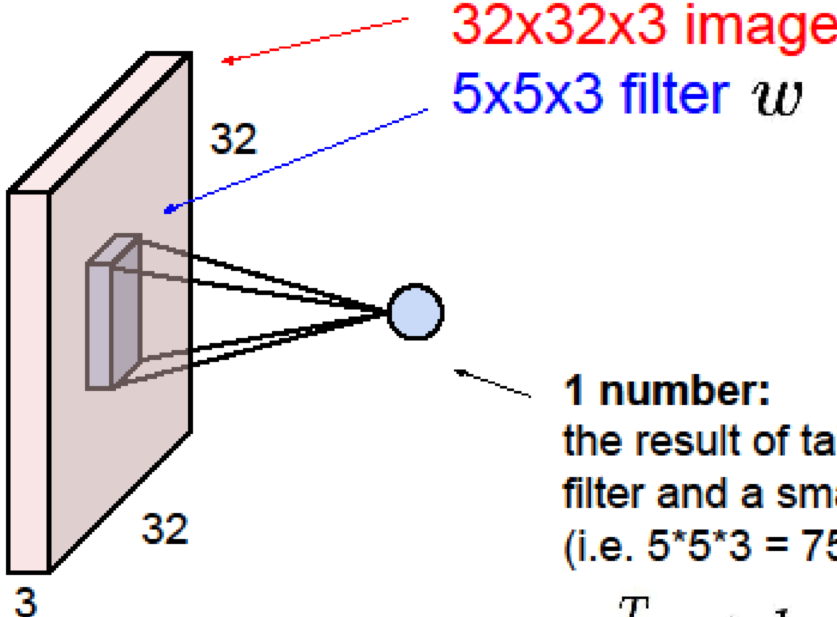


Filters always extend the full depth of the input volume

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

One Conv Layer

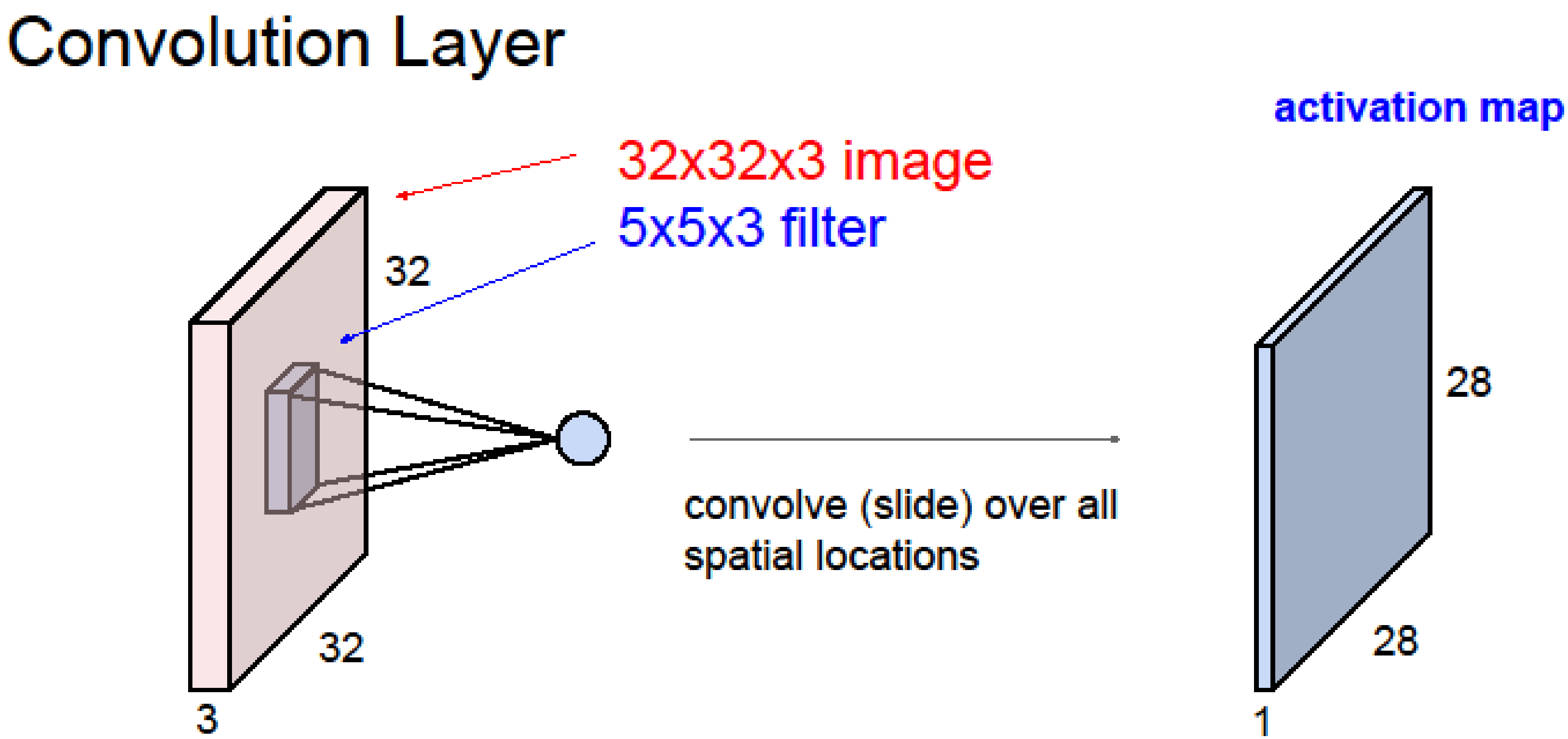
Convolution Layer



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)

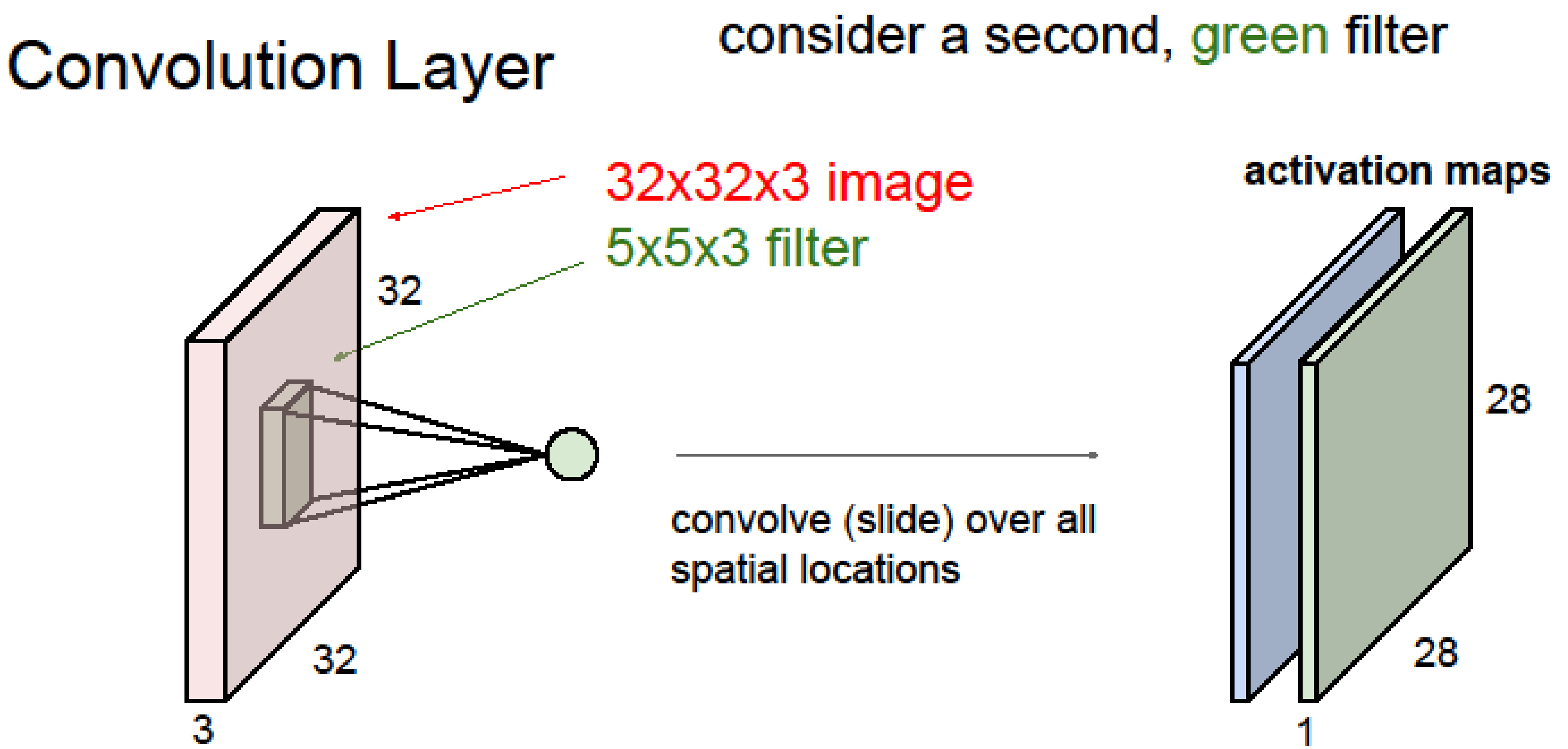
$$w^T x + b$$

One Conv Layer



Question: What are the values of W, P, F, and S in this example?

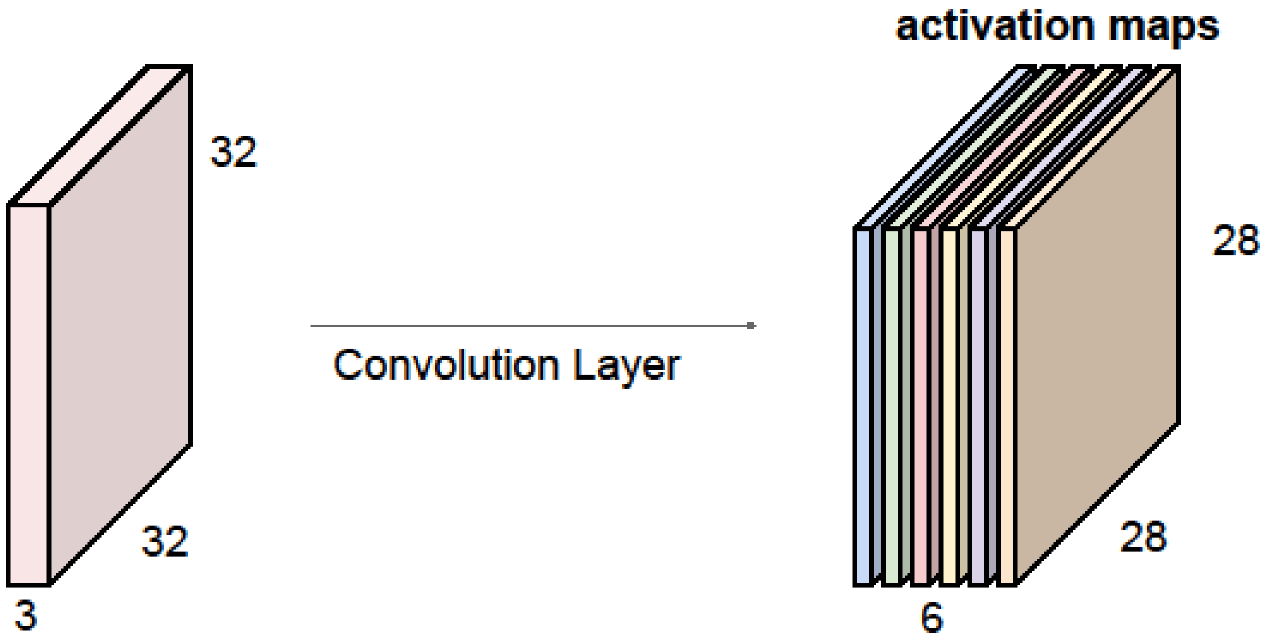
One Conv Layer



One Conv Layer

Also called:
feature map, or
depth slice

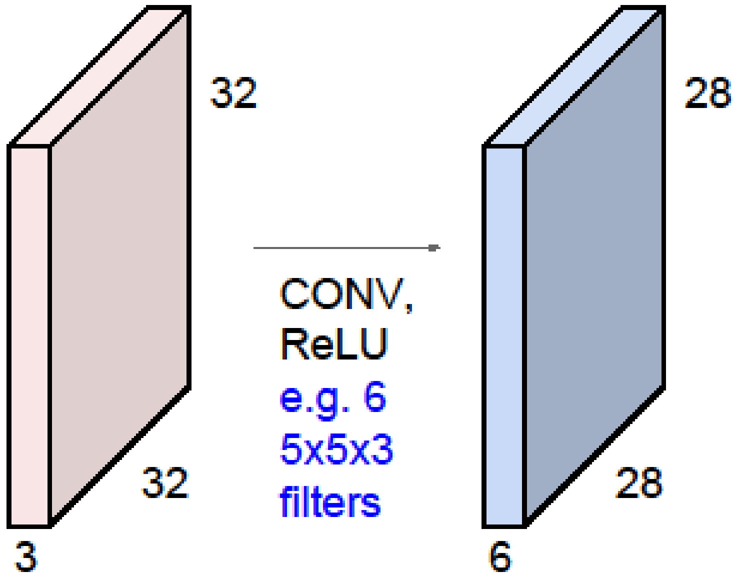
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

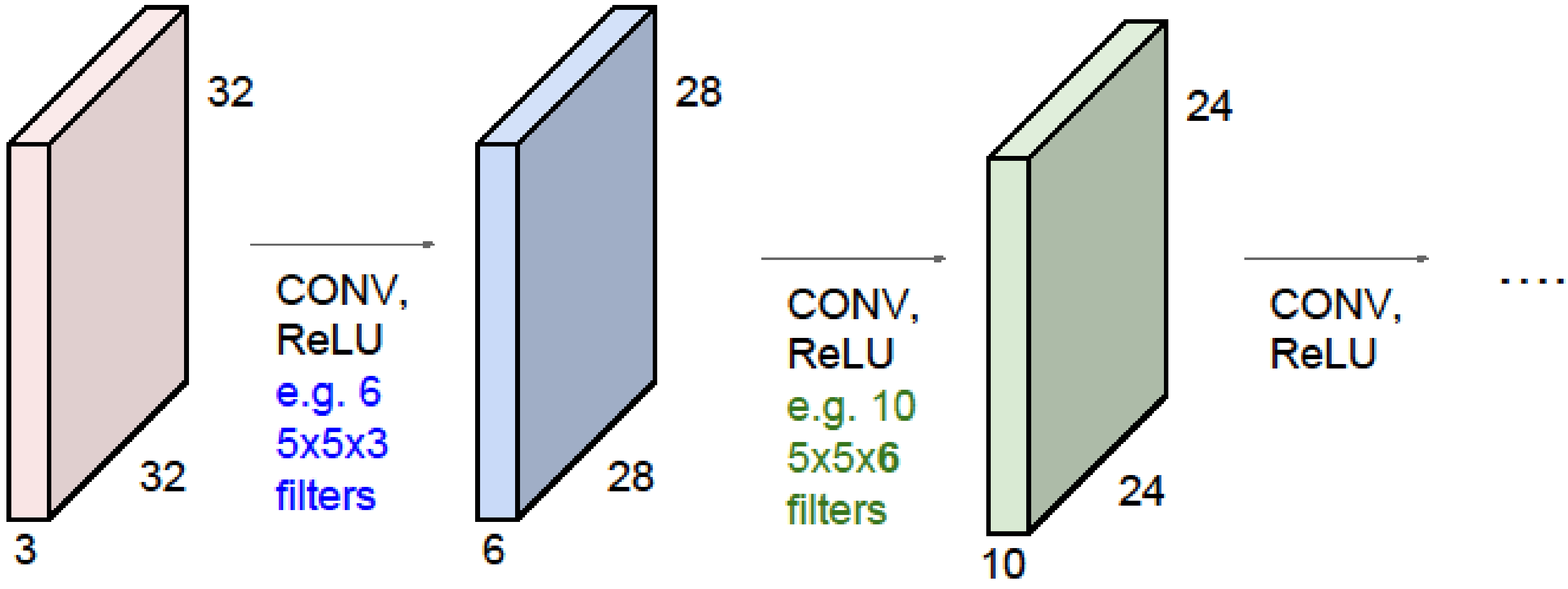
Convolutional Network (ConvNet)

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



Convolutional Network (ConvNet)

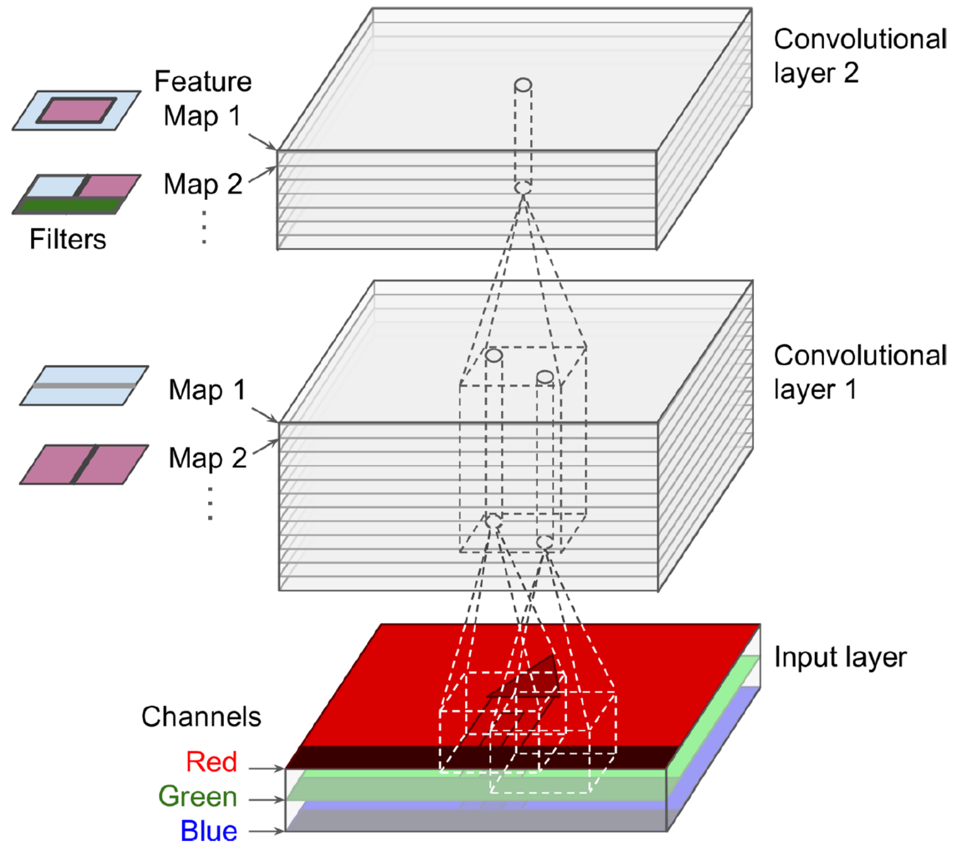
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Convolutional Layer

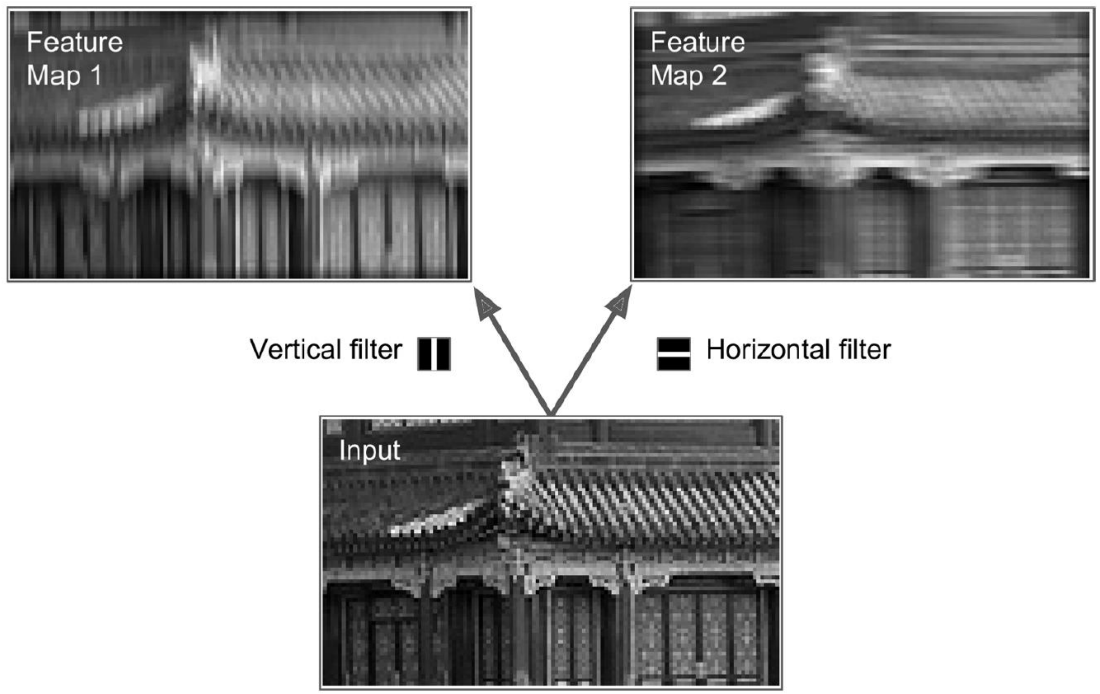
- Neurons in the 1st conv. Layer
  - not connected to every single pixel in the input image
  - only connected to pixels in their own receptive fields

- Neurons in the 2nd conv. Layer
  - connected only to neurons located within a small cuboid in the 1st layer



Filters

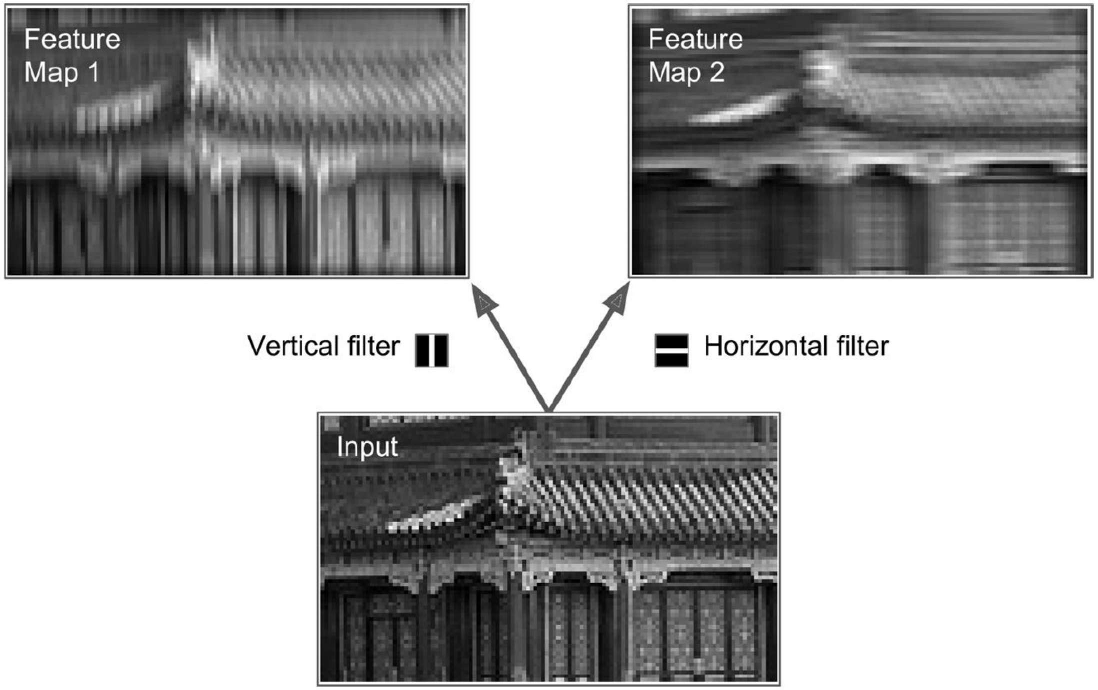
- A filter can be represented as: a small image the size of the receptive field



  – Apply two different filters to the same image, to get two feature maps

Filters

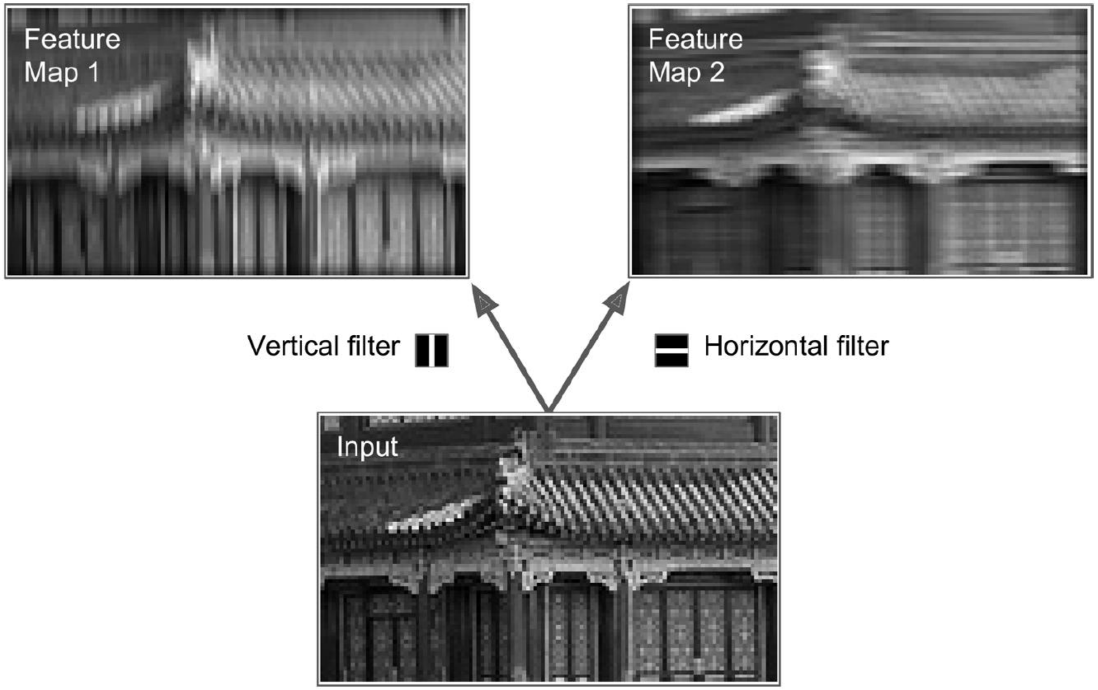
- If all neurons in a layer use the same vertical line filter (and the same bias term)



- In the output feature map: the vertical white lines get enhanced, while the rest gets blurred

Filters

- If all neurons in a layer use the same horizontal line filter (and the same bias term)



- In the output feature map: the horizontal white lines get enhanced, while the rest is blurred out

Filters

- A layer full of neurons using the same filter gives you a feature map, which highlights the areas in an image that are most similar to the filter's pattern
- During training, a CNN finds the most useful filters (i.e. filter patterns) for its task (such as classification, or regression), and it learns to combine them into more complex patterns

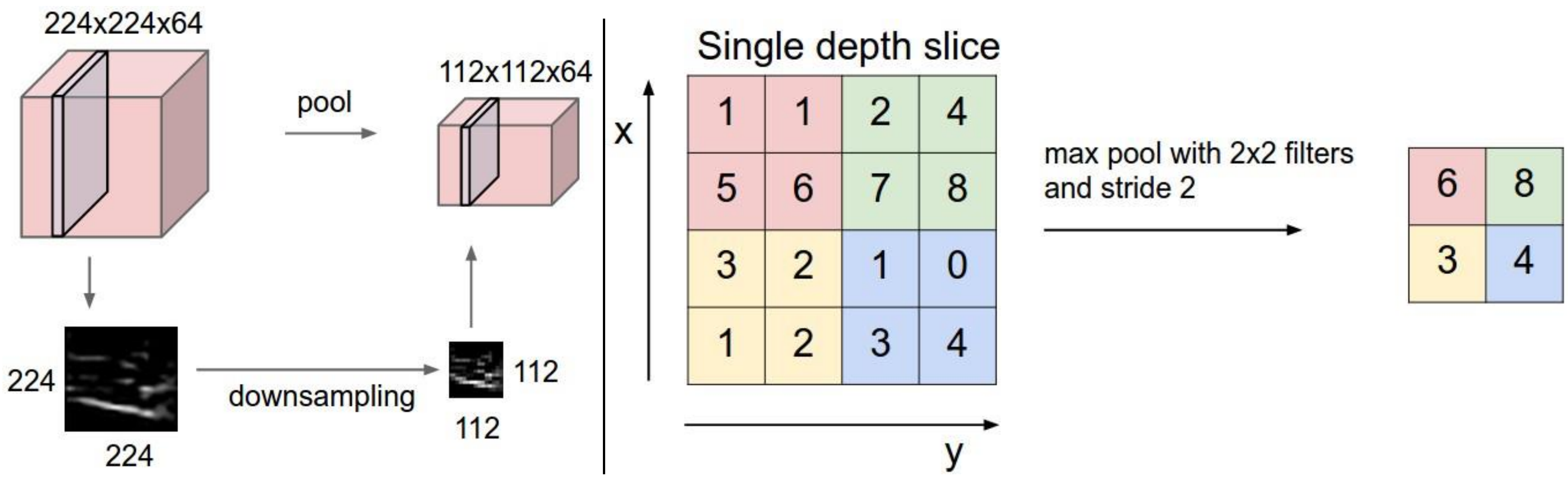
Summary

- One convolutional layer: composed of several 2D feature maps of equal sizes, therefore one such layer is 3D
- All neurons within one feature map: share the same filter (i.e. parameters/weights)
  - The elements in the filter are the parameters/weights
- Different feature maps may have different parameters

- A convolutional layer: simultaneously applies multiple filters to its inputs, making it capable of detecting multiple features in its inputs

CNN-Pooling Layer

- Max pooling



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

Pooling Layer

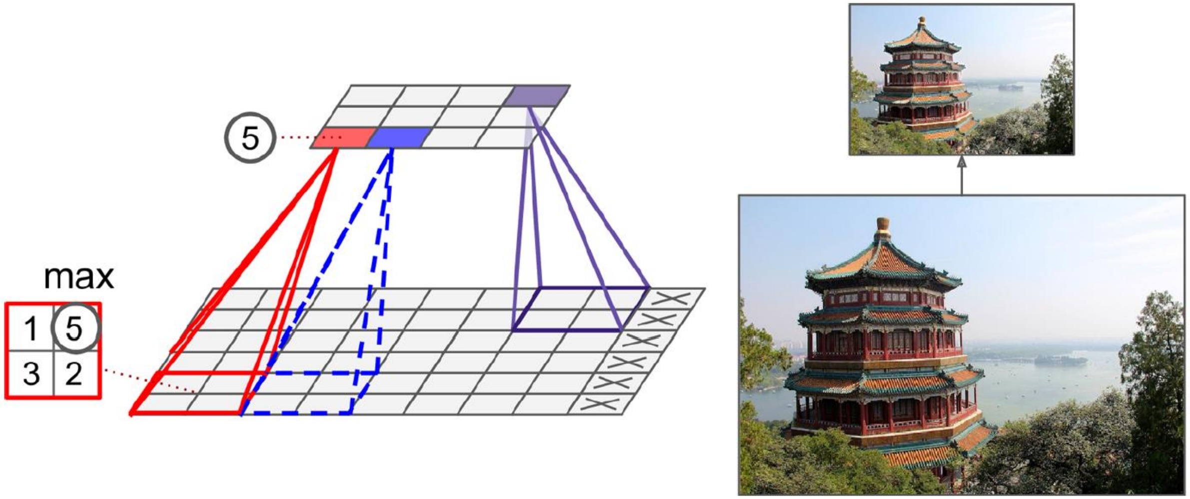
- Subsample the input image in order to reduce
  - the computational load
  - The memory use
  - The number of parameters (thereby limiting the risk of overfitting)
- Each neuron in a pooling layer is generated by using a limited number of neurons in the previous layer

Pooling Layer

- A pooling neuron has no weights
- It aggregates the inputs using an aggregation function
  - Max
  - Mean
- Works on every input channel independently
- Output depth (number of channels): the same as the input depth

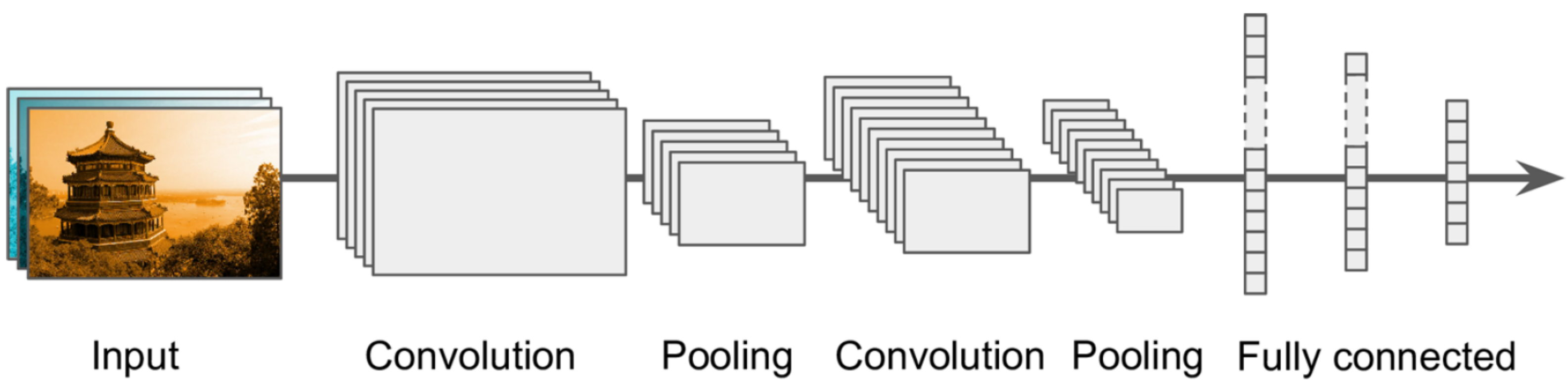
Pooling Layer

- A max pooling layer, with
  - 2×2 pooling kernel
  - A stride of 2
  - No padding



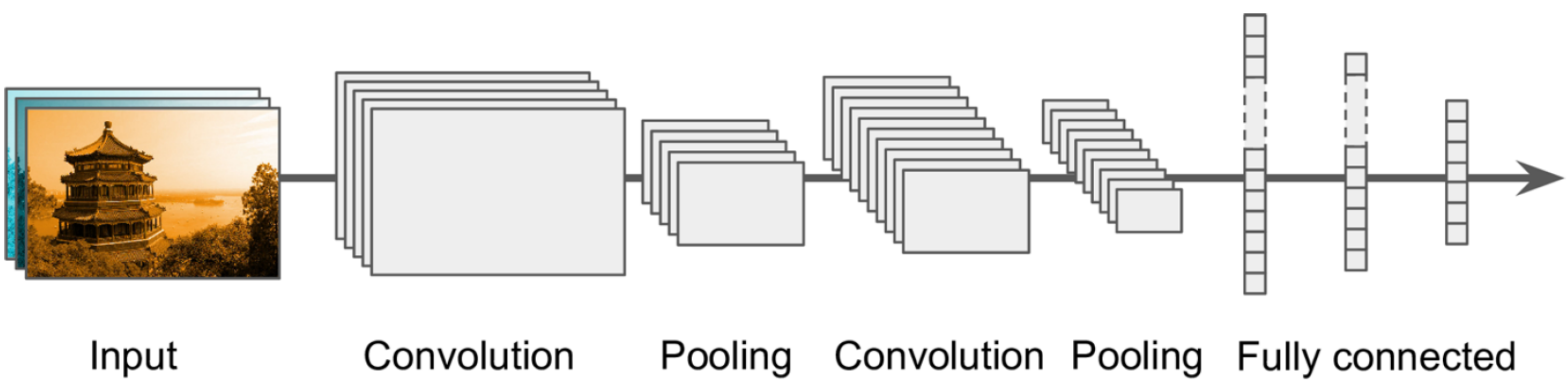
Typical CNN Architecture

- One convolutional layer
  - Multiple feature maps
  - Followed by a ReLU operation
- Then a pooling layer



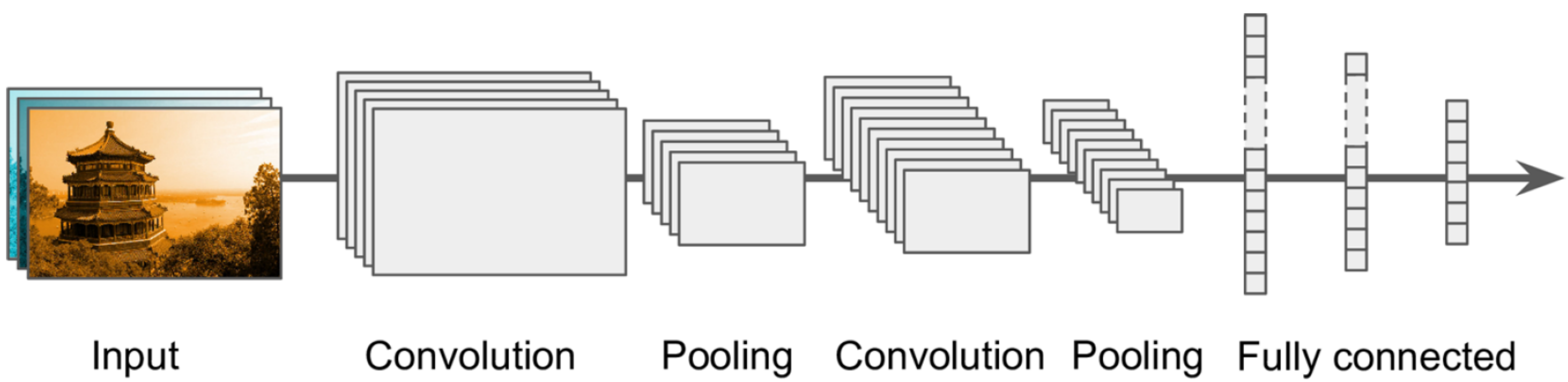
Typical CNN Architecture

- Then another convolutional layer
- Then another pooling layer
- And so on…



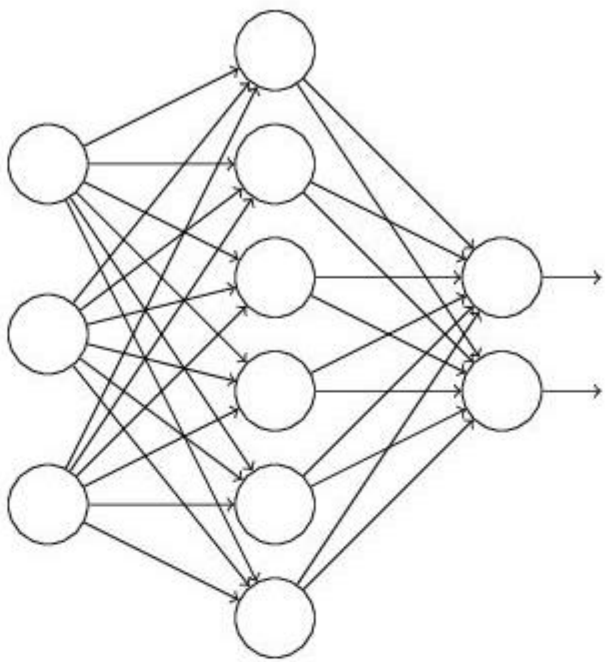
Typical CNN Architecture

- Then a regular neural network
  - A few fully connected layers
  - Each followed by a ReLU
  - The final layer outputs the prediction (a softmax layer that outputs estimated class probabilities)

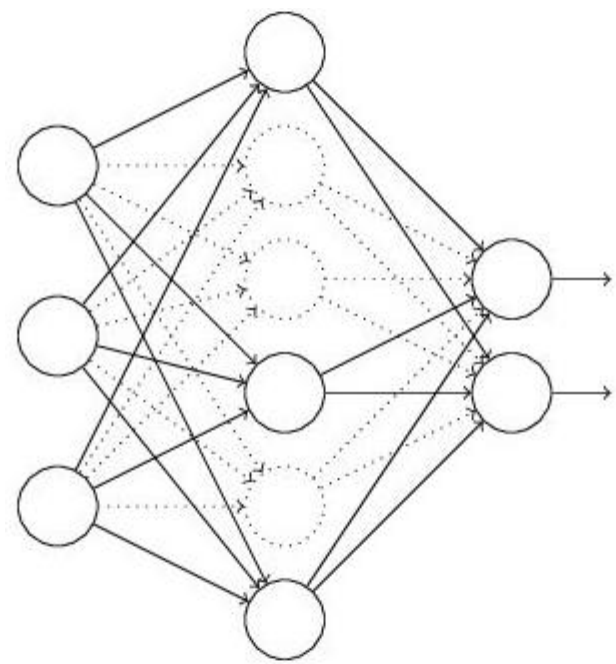


Dropout Layer

- Address the problem of over-fitting, while
  - Network goes deeper
  - Layer goes larger



Before dropout



After dropout

ImageNet

- A dataset of over 15 million labeled high-resolution images
- Roughly 22,000 categories
- The images were collected from the web and labeled by human labelers

ImageNet Challenge

- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)
  - An annual competition (2010-2017)
  - Uses a subset of ImageNet database
  - 1,000 categories, roughly 1,000 images each
    - Some of the categories are really subtle (trying to distinguish 120 dog breeds)
  - Roughly 1.2 million training images
  - 50,000 validation images
  - 150,000 test images

ImageNet Challenge

- Tasks
  - Image Classification
  - Single-Object Localization
  - Object Detection

ImageNet Challenge Tasks