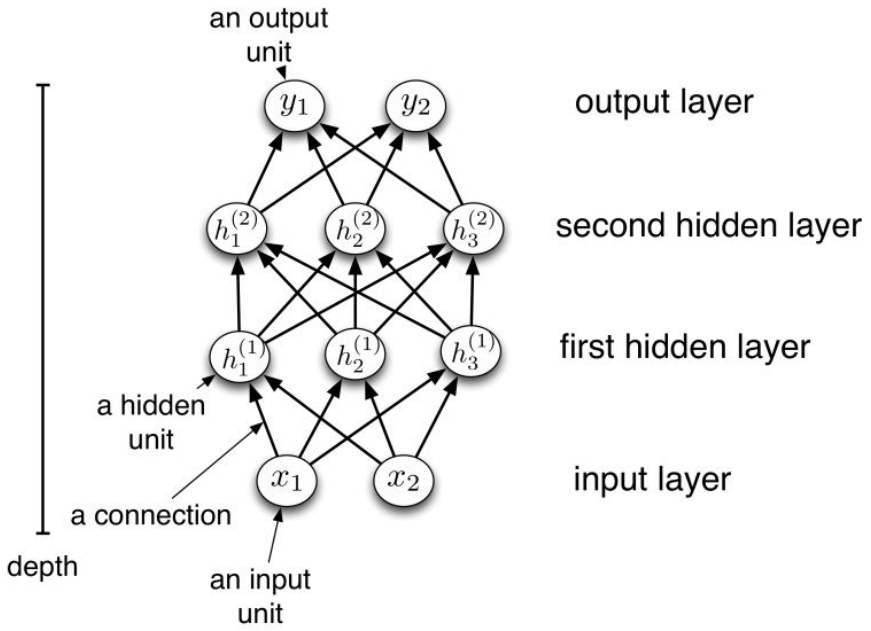
CHL7001H S1 Applied Deep Learning

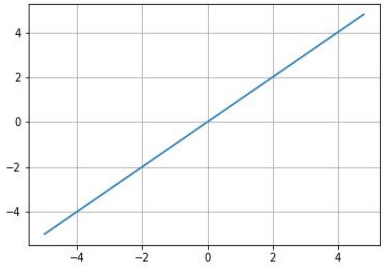
Modern Deep Learning

Recap of last week

- Fully connected layers (e.g. Multilayer Perceptrons)
- In the simplest case, all input units are connected to all output units. We call this a **fully connected layer**.
- A multilayer network consisting of fully connected layers is called a **multilayer perceptron**.

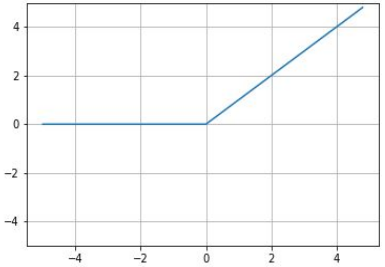


Some activation functions



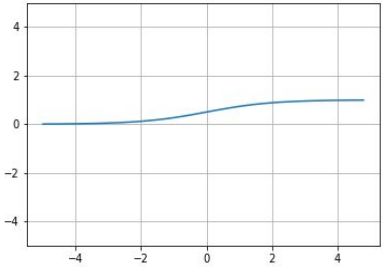
Identity

$$f(x) = x$$



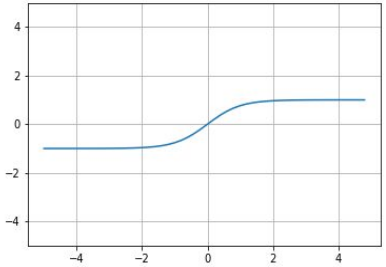
ReLU

$$f(x) = max(0, x)$$



Logistic

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



TanH

$$f(x) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

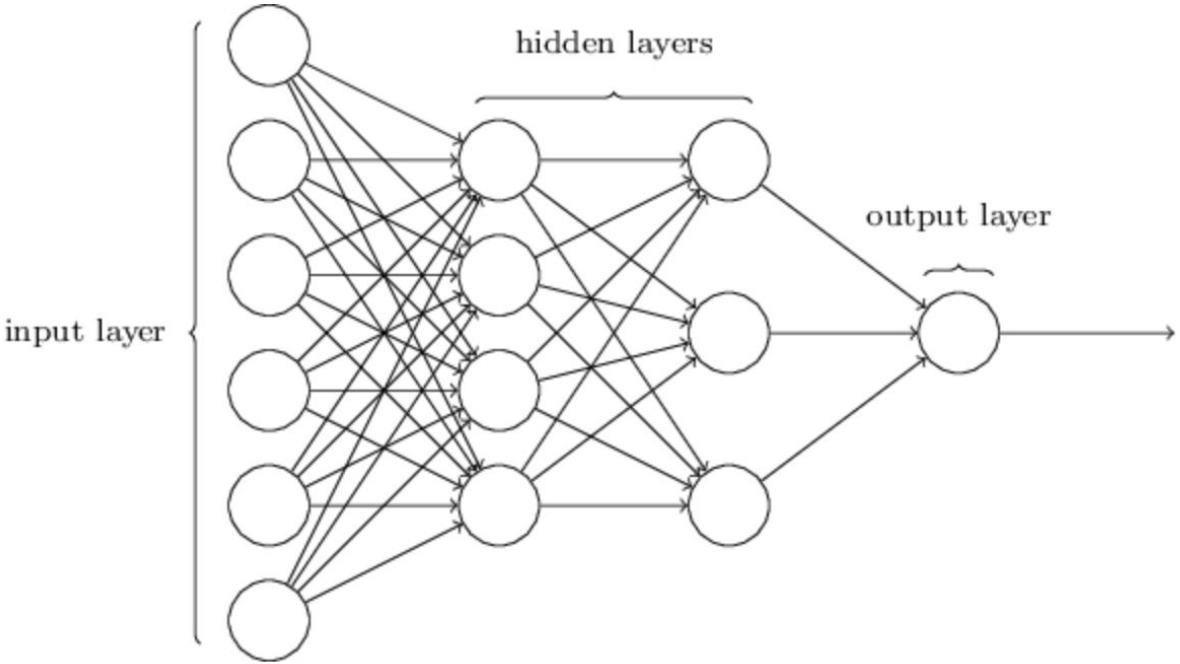
Example of shallow NN

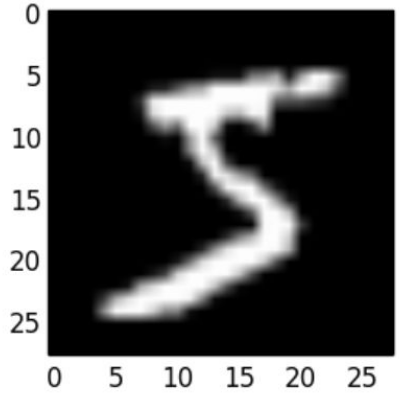
- https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=8&seed=0.50265&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

Example with a deep NN

- https://playground.tensorflow.org/#activation=relu&batchSize=30&dataset=spiral&regDataset=reg-plane&learningRate=0.1&regularizationRate=0&noise=0&networkShape=4,4,4&seed=0.50265&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

Handwritten digits classification



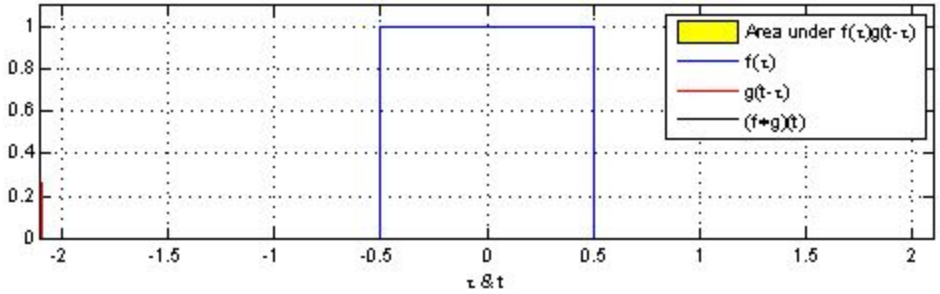


Problem with fully connected network

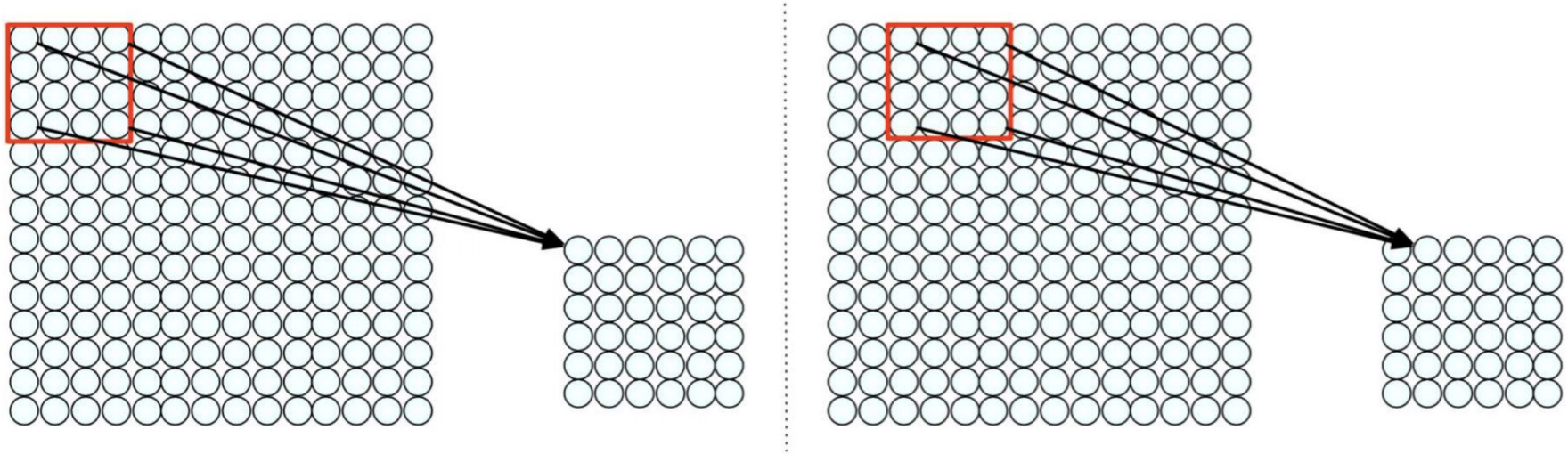
What is the problem with having this as the architecture (densely connected)?

- Too many parameters!
  - Input: 28*28 = 784
  - If we plan to have 100 hidden units, then the number of parameters will be 784*100 = 78.4K
- What happens if the object in the image shifts a little?
- No spatial information!

Example of convolution

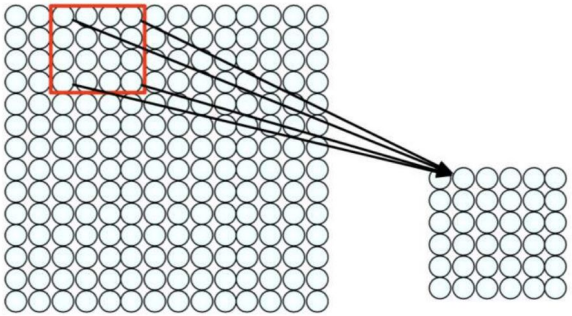


Using spatial structure



- Connect patch in input layer to a single neuron in subsequent layer.
- Use a sliding window to define connections.

Feature extraction with convolution



- Filter of size 4x4 : 16 different weights
- Apply this same filter to 4x4 patches in input
- Shift by 2 pixels for next patch

This “patchy” operation is convolution

1. Apply a set of weights – a filter – to extract local features
2. Use multiple filters to extract different features - sliding
3. Spatially share parameters of each filter or say weights are shared between all image location

Feature maps

Original

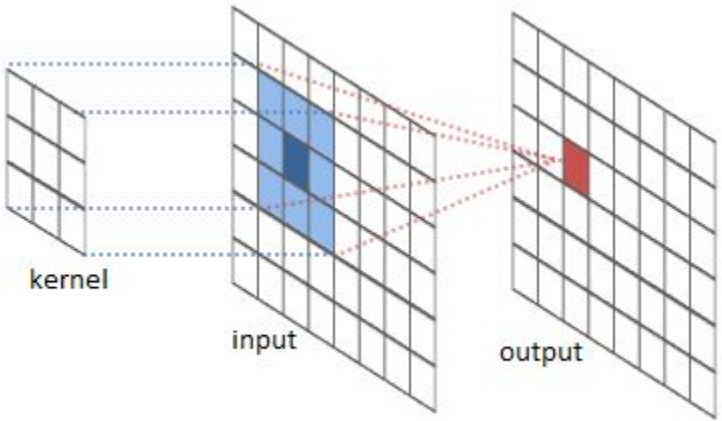
Sharpen

Edge Detect

"Strong" Edge Detect

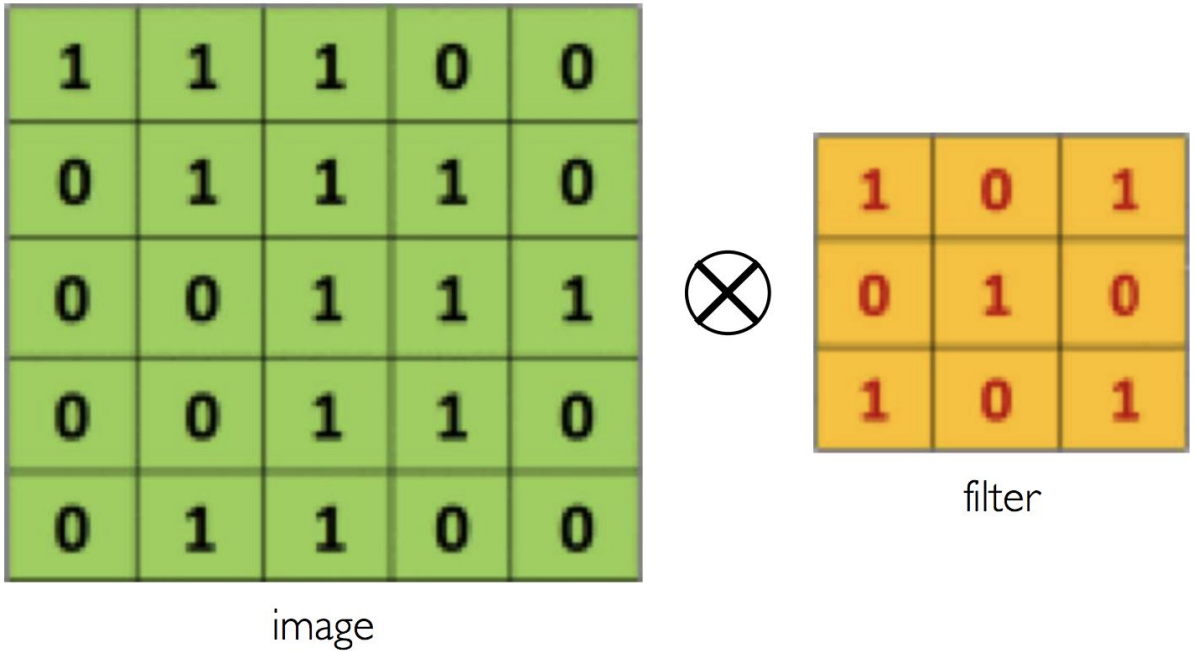
Source: MIT 6S191

Convolution operation



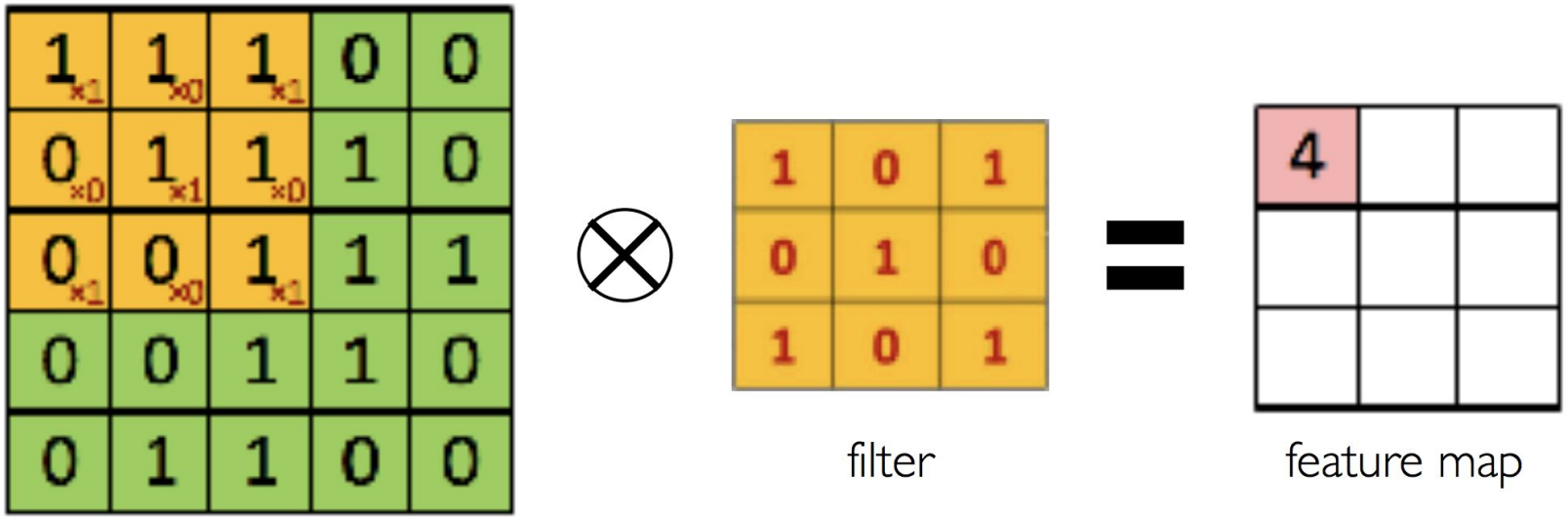
Convolution operation

Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:

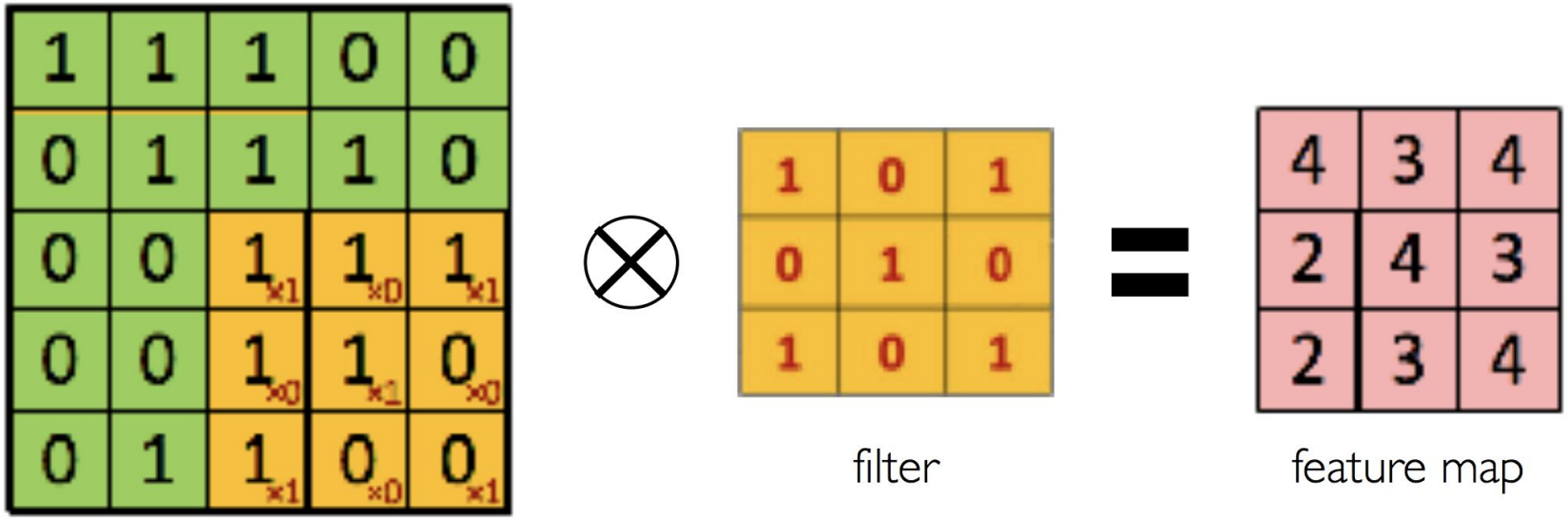


Convolution operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



Convolution operation



Simple filters

0	0	0	0	0
0	1/9	1/9	1/9	0
0	1/9	1/9	1/9	0
0	1/9	1/9	1/9	0
0	0	0	0	0

Source: MIT 6S191

Simple filters

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

Source: MIT 6S191

Feature maps

Original

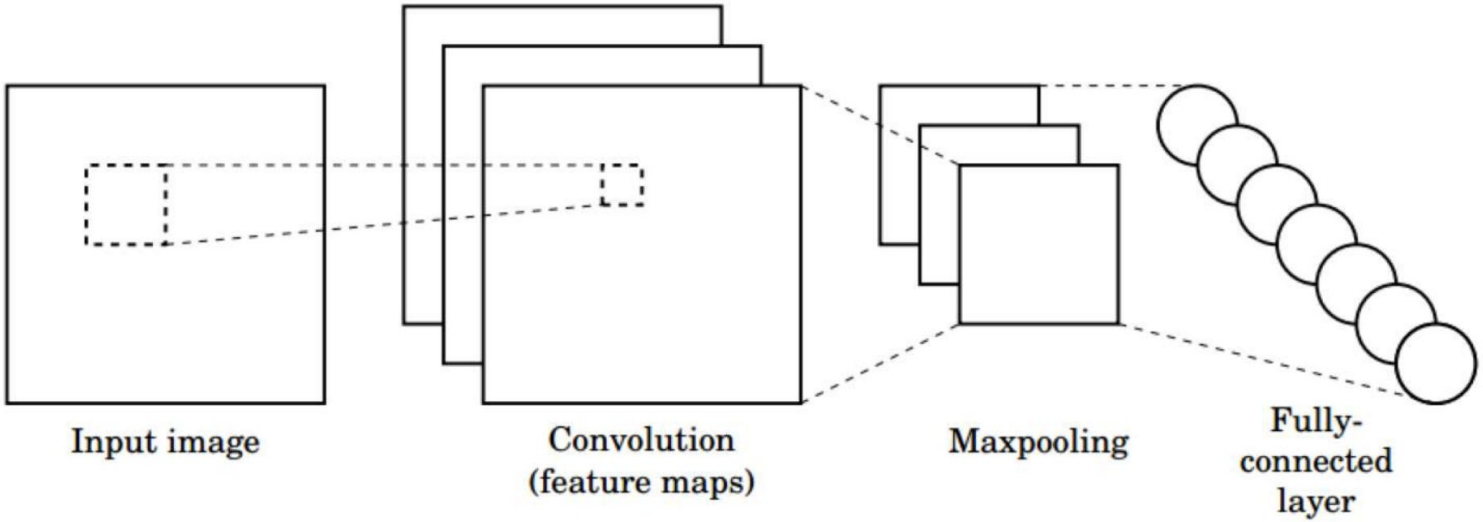
Sharpen

Edge Detect

"Strong" Edge Detect

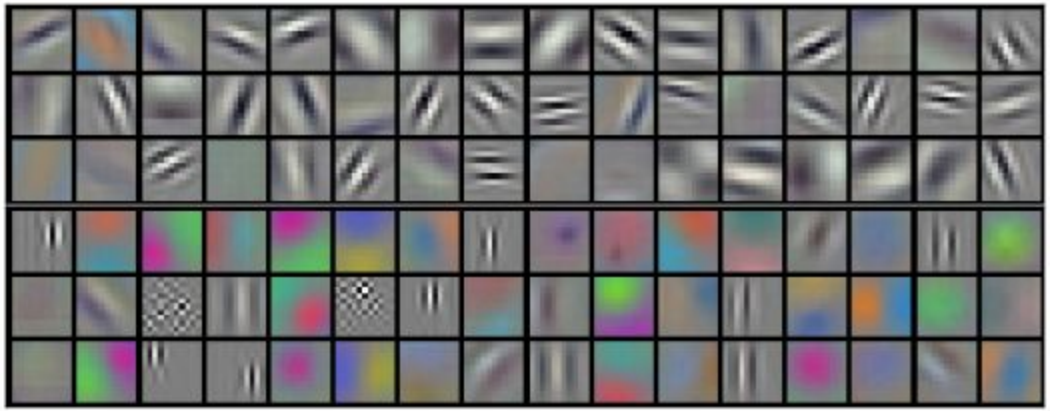
Source: MIT 6S191

CNN for classification



1. Convolution: Apply filters with learned weights to generate feature maps.
2. 2. Non-linearity: Often ReLU. 3. Pooling: Downsampling operation on each feature map.

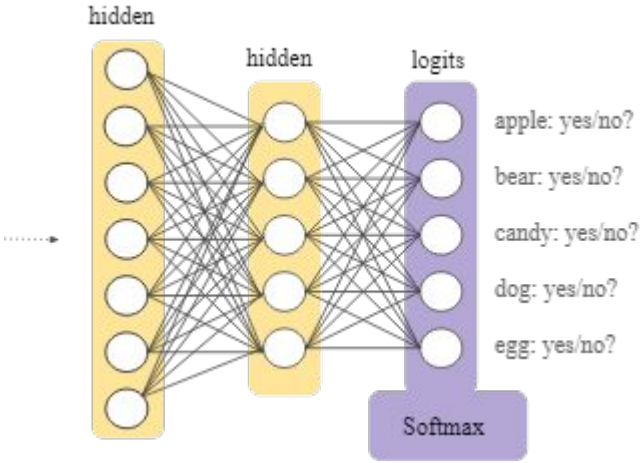
CNN for classification



1. Convolution: Apply filters with learned weights to generate feature maps.
2. 2. Non-linearity: Often ReLU. 3. Pooling: Downsampling operation on each feature map.

Multiclass

- Multiclass, single label? Softmax
- Multiclass, multilabel? Sigmoids



Examples

- https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/r2/tutorials/quickstart/beginner.ipynb
- https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/r2/tutorials/images/intro_to_cnns.ipynb