DS 4400

Machine Learning and Data Mining I
Spring 2022

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

April 11 2022

Announcements

- Project milestone is due on April 13
  - Template in Gradescope
  - We would like to see at least one trained ML model
  - Discuss any challenges
- Final exam is on Wed, April 20
  - During class
  - Same format as the midterm
  - Topics – everything from LDA to CNN

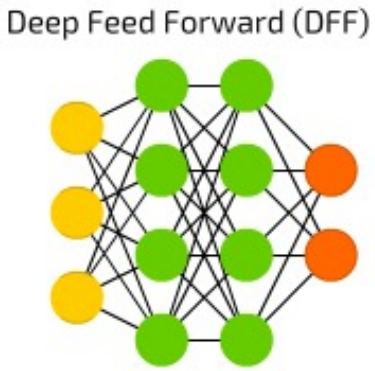
Outline

- Feed-forward neural networks
  - Activations
  - Softmax classifier
  - Architectures and parameters
- Convolutional neural networks
  - Convolution layer
  - Max pooling
  - Well-known convolutional networks architectures

Neural Network Architectures

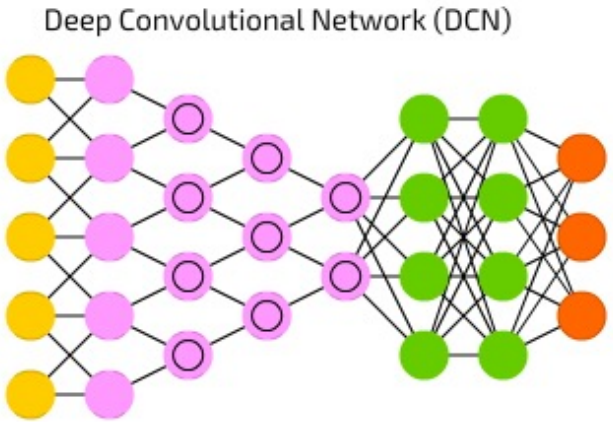
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



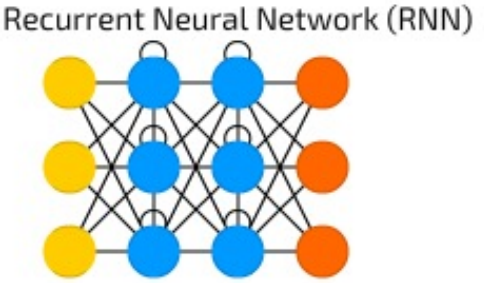
Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations



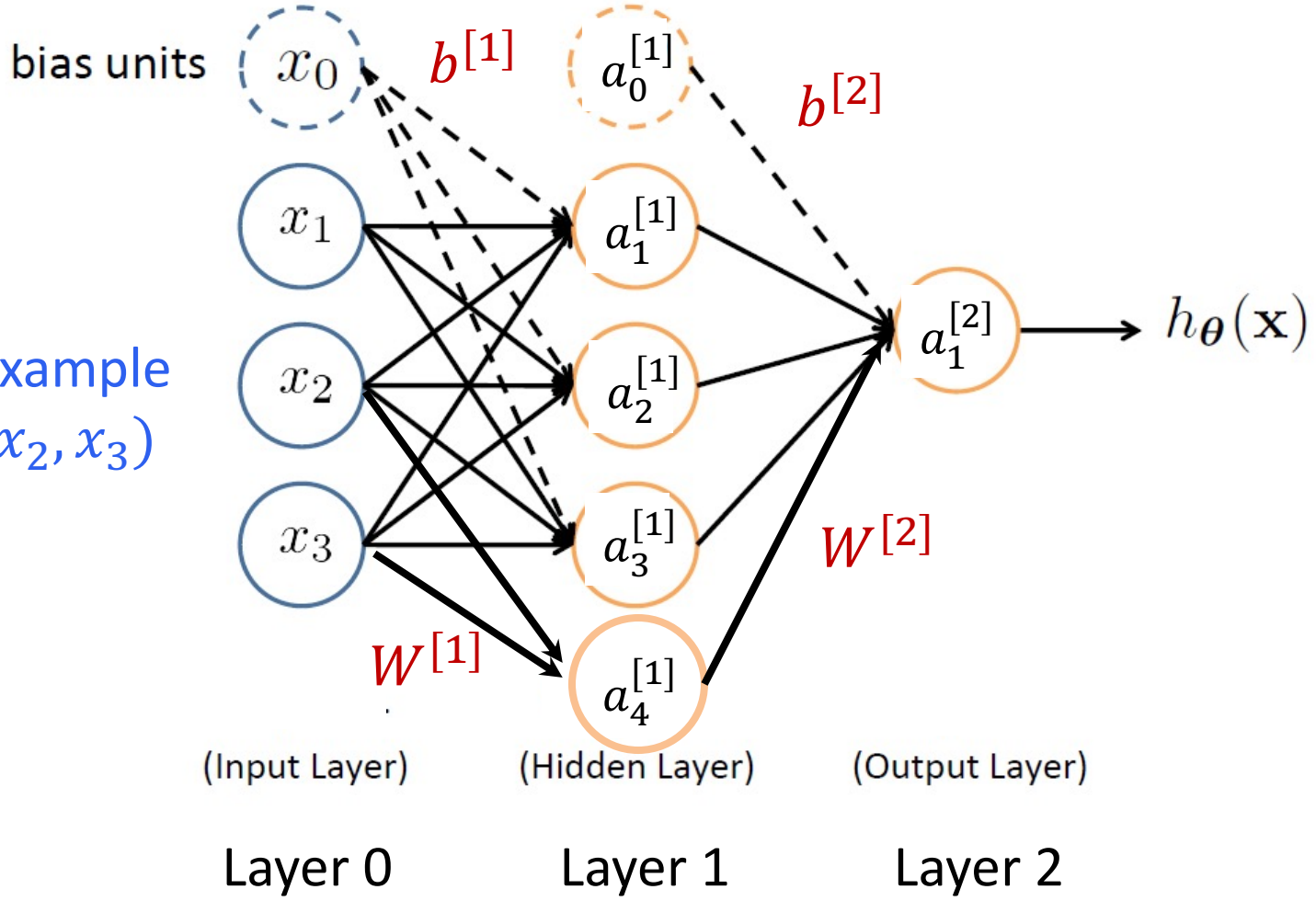
Recurrent Networks

- Keep hidden state
- Have cycles in computational graph



Feed-Forward Neural Network

Training example
$x = (x_1, x_2, x_3)$



Feed-Forward NN

- Hyper-parameters
  - Number of layers
  - Architecture (how layers are connected)
  - Number of hidden units per layer
  - Number of units in output layer
  - Activation functions
- Other
  - Initialization
  - Regularization

Vectorization

$$z_1^{[1]} = W_1^{[1]}\ x + b_1^{[1]} \quad \text{and} \quad a_1^{[1]} = g(z_1^{[1]})$$
$$\vdots \qquad\qquad\qquad \vdots \qquad\qquad \vdots$$
$$z_4^{[1]} = W_4^{[1]}\ x + b_4^{[1]} \quad \text{and} \quad a_4^{[1]} = g(z_4^{[1]})$$

$$\underbrace{\begin{bmatrix} z_1^{[1]} \\ \vdots \\ z_4^{[1]} \end{bmatrix}}_{z^{[1]} \in \mathbb{R}^{4\times 1}} = \underbrace{\begin{bmatrix} — & W_1^{[1]} & — \\ — & W_2^{[1]} & — \\ & \vdots & \\ — & W_4^{[1]} & — \end{bmatrix}}_{W^{[1]} \in \mathbb{R}^{4\times 3}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x \in \mathbb{R}^{3\times 1}} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_4^{[1]} \end{bmatrix}}_{b^{[1]} \in \mathbb{R}^{4\times 1}}$$

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

Linear

$$a^{[1]} = g(z^{[1]})$$

Non-Linear

Vectorization

Output layer

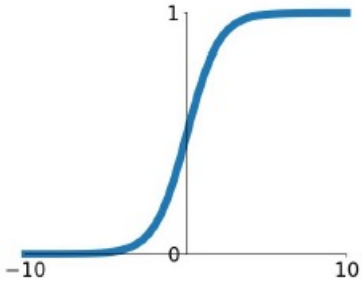
$$z_1^{[2]} = W_1^{[2]^T} a^{[1]} + b_1^{[2]} \quad \text{and} \quad a_1^{[2]} = g(z_1^{[2]})$$

$$\underbrace{z^{[2]}}_{1\times1} = \underbrace{W^{[2]}}_{1\times4}\underbrace{a^{[1]}}_{4\times1} + \underbrace{b^{[2]}}_{1\times1} \quad \text{and} \quad \underbrace{a^{[2]}}_{1\times1} = g(\underbrace{z^{[2]}}_{1\times1})$$

Non-Linear Activation Functions

Sigmoid

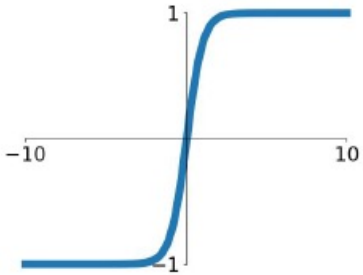
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Binary Classification

tanh

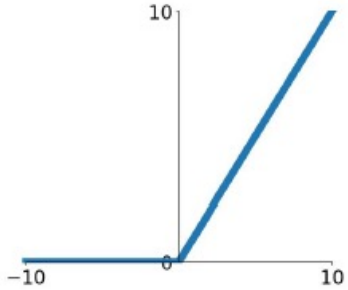
$$\tanh(x)$$



Regression

ReLU

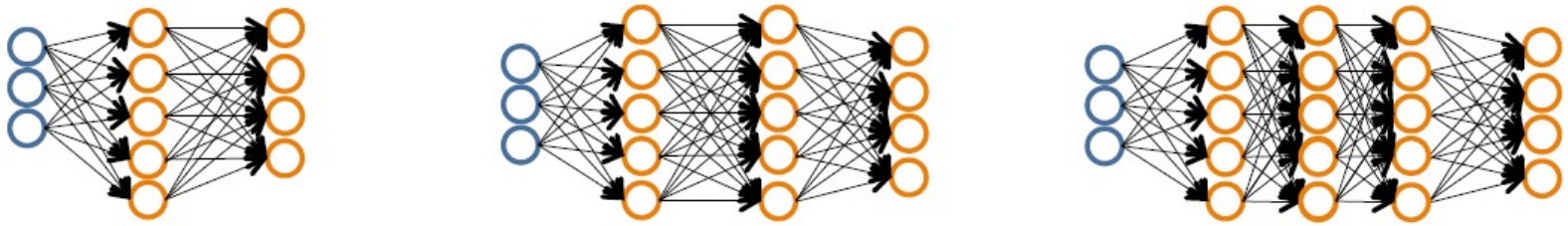
$$\max(0, x)$$



Intermediary layers

How to pick architecture?

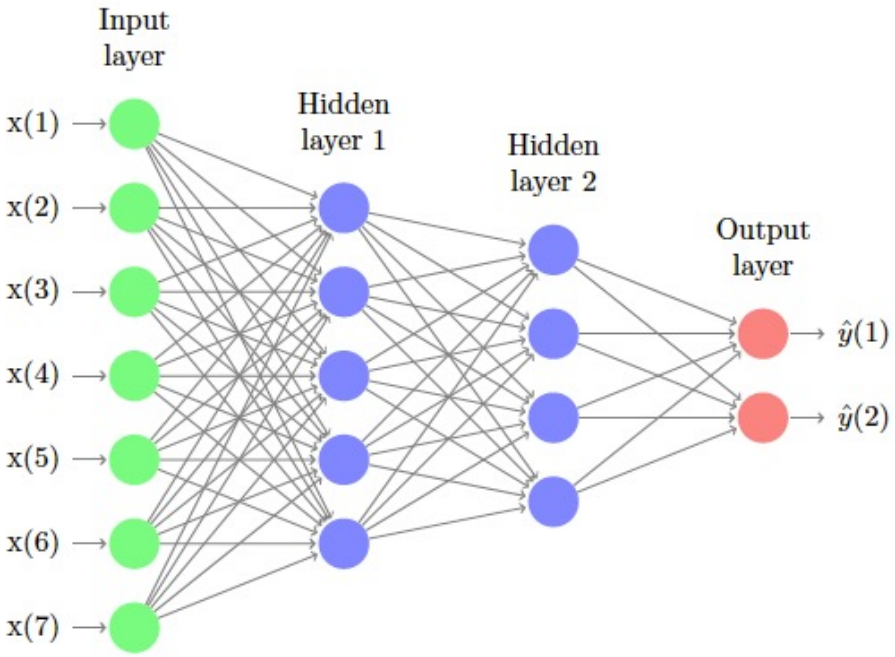
Pick a network architecture (connectivity pattern between nodes)



- # input units = # of features in dataset
- # output units = # classes

Reasonable default: 1 hidden layer

FFNN Architectures



- Input and Output Layers are completely specified by the problem domain
- In the Hidden Layers, number of neurons in Layer i+1 is usually smaller or equal to the number of neurons in Layer i

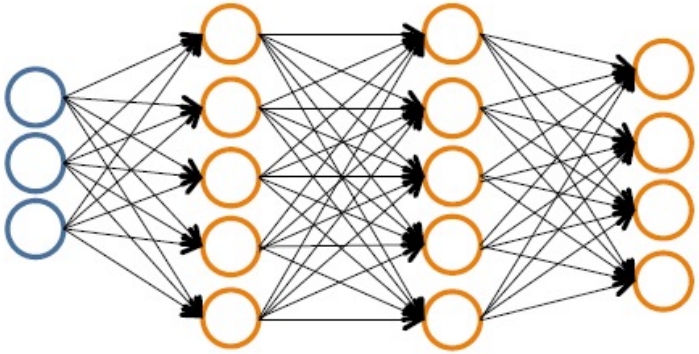
Multi-Class Classsification

Pedestrian

Car

Motorcycle

Truck



$$h_\Theta(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

when pedestrian

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

when car

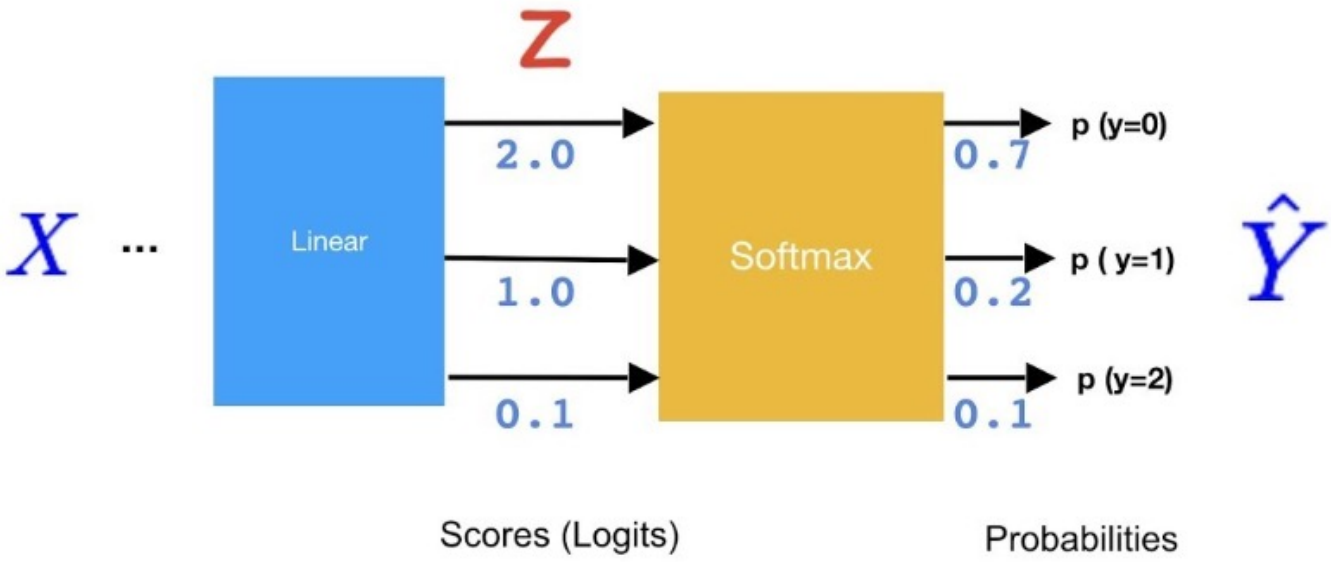
$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

when motorcycle

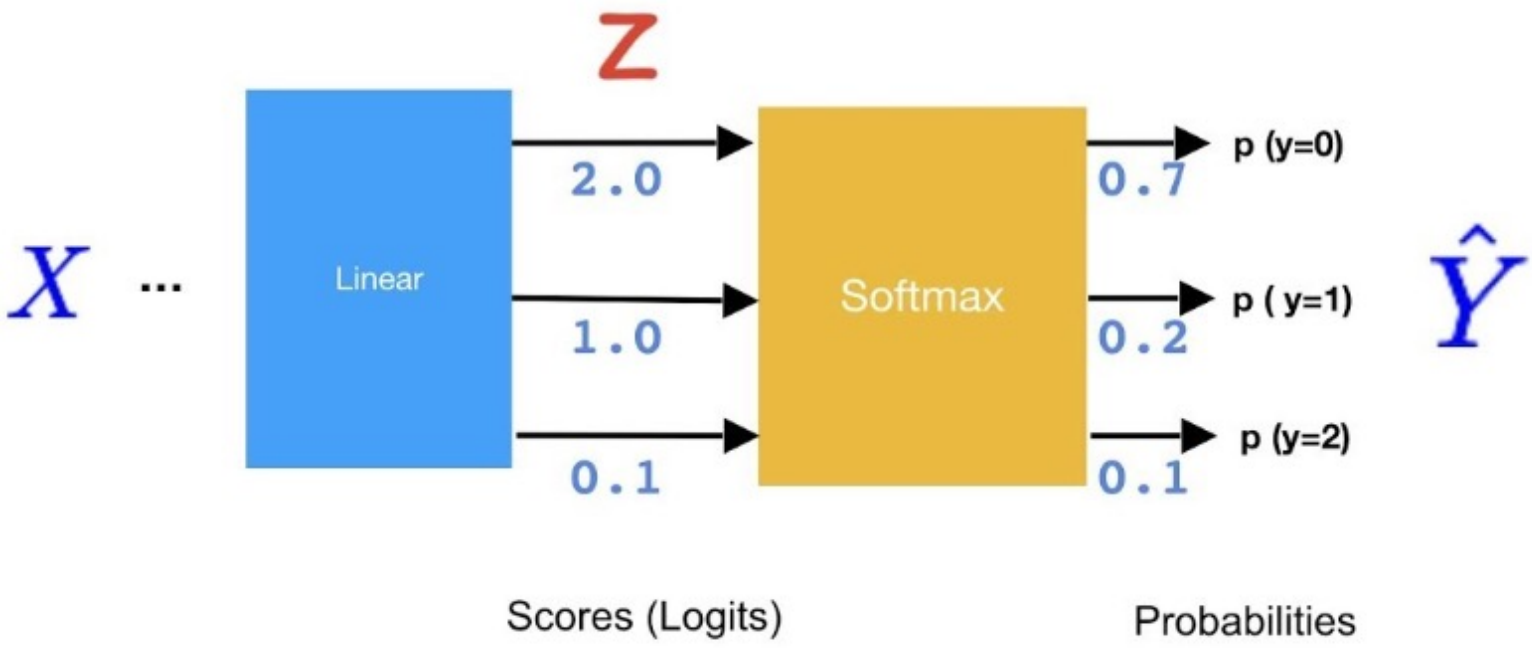
$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when truck

Softmax classifier



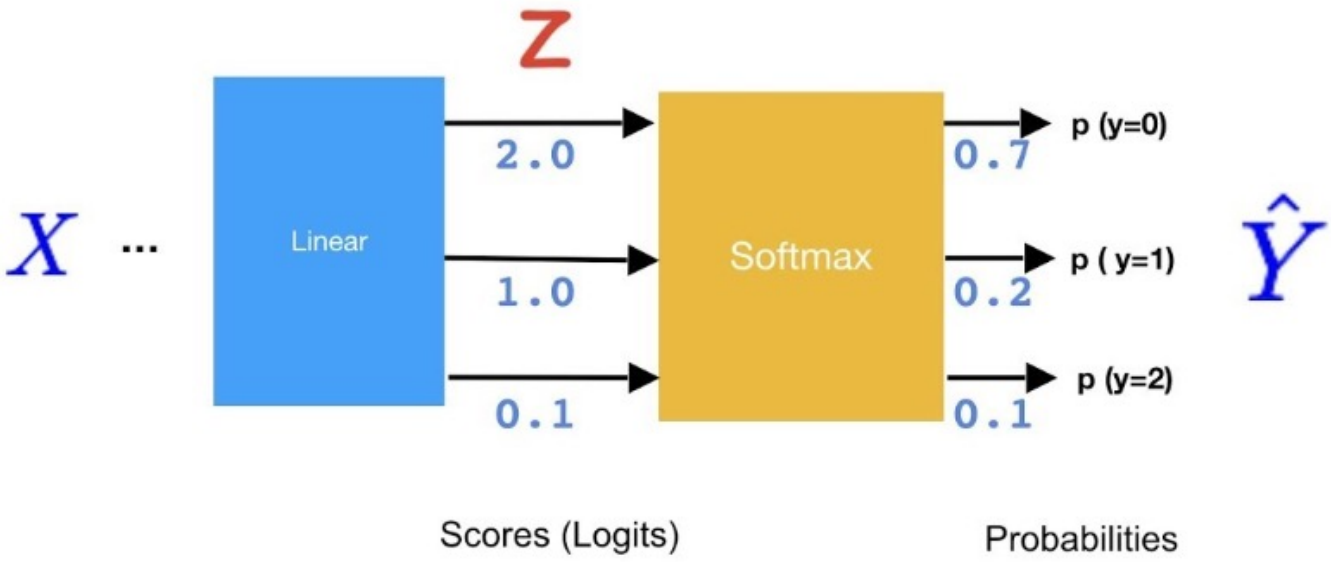
Softmax classifier



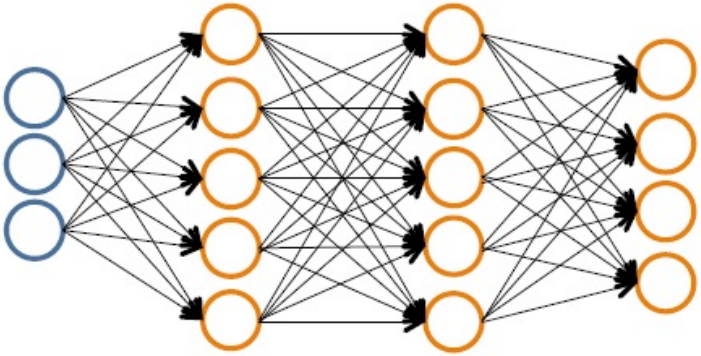
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

- Predict the class with highest probability
- Generalization of sigmoid/logistic regression to multi-class

Cross-entropy loss



Neural Network Classification



Given:

$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

$\mathbf{s} \in \mathbb{N}^{+L}$ contains # nodes at each layer

- $s_0 = d$ (# features)

Binary classification

y = 0 or 1

1 output unit $(s_{L-1} = 1)$

Sigmoid

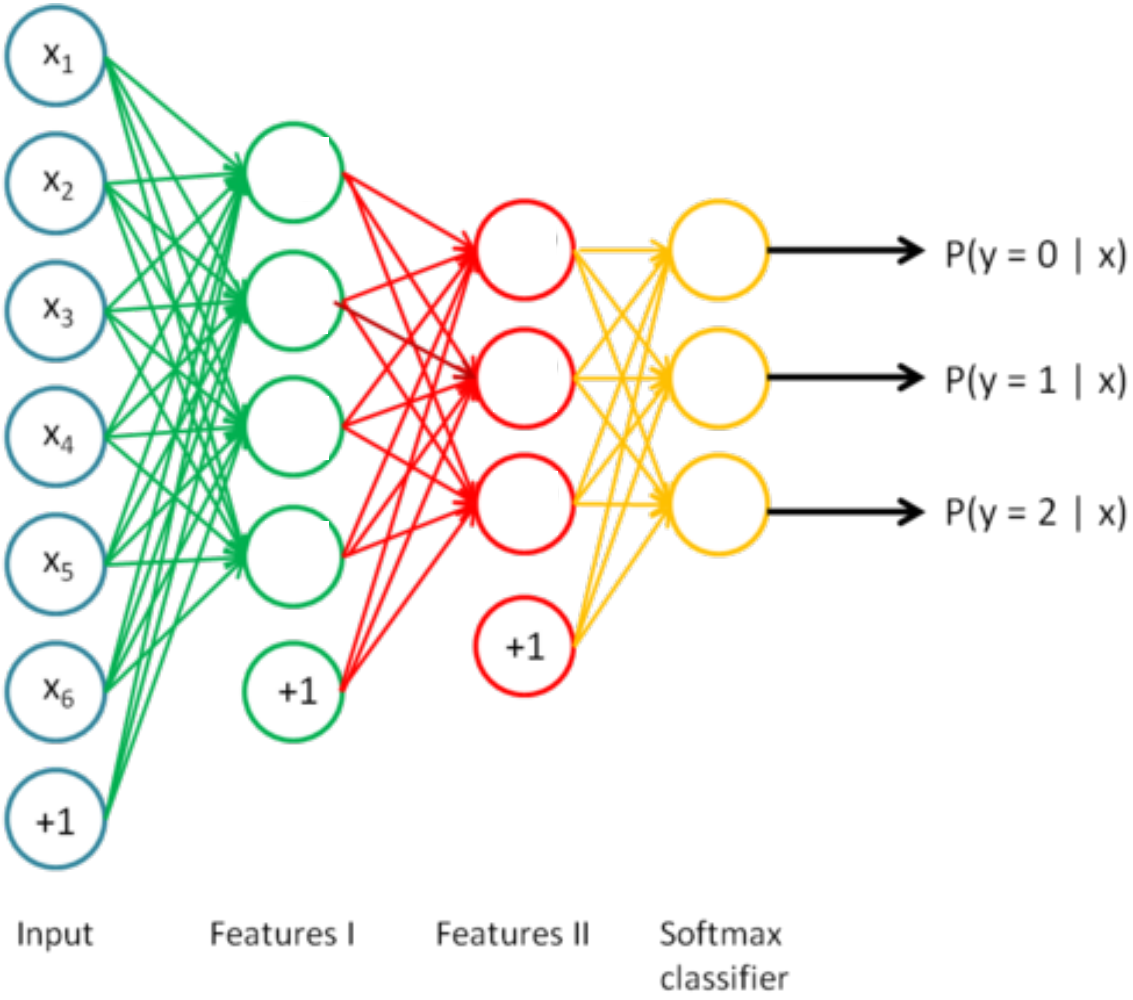
Multi-class classification (K classes)

$\mathbf{y} \in \mathbb{R}^K$ e.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ pedestrian, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ car, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$ motorcycle, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$ truck

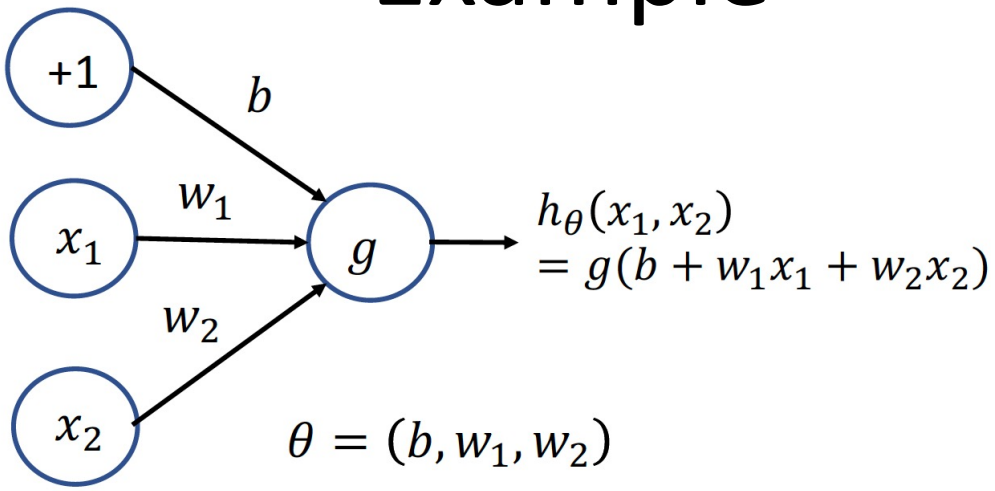
K output units $(s_{L-1} = K)$

Softmax

Multi-class classification



Example



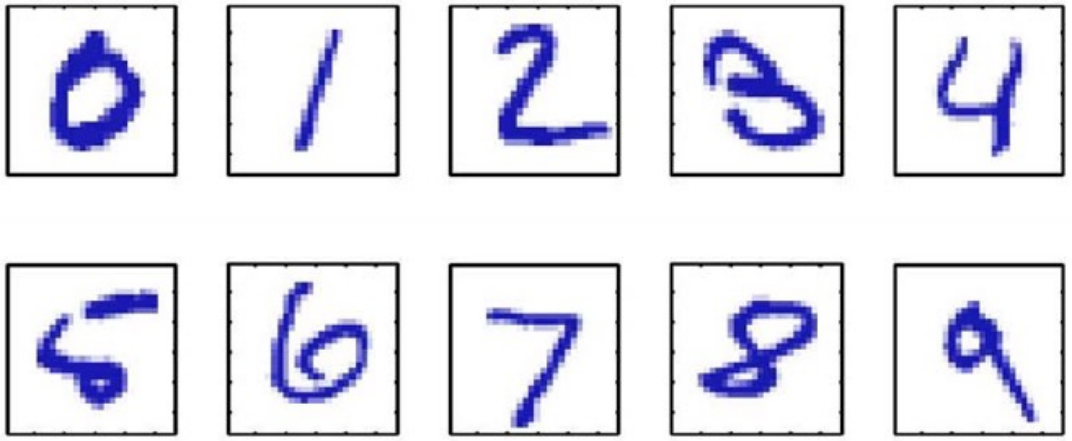
1. Given $b = -10, w_1 = 12, w_2 = 5$
   Activation $g(z) = sign(z)$
Compute the output:

x_1	x_2	$h(x_1, x_2)$
0	0	
0	1	
1	0	
1	1	

2. Find out the weights b, w_1, w_2 and activation function to get the following output:

x_1	x_2	$h(x_1, x_2)$
0	0	1
0	1	1
1	0	1
1	1	0

MNIST: Handwritten digit recognition



Images are 28 x 28 pixels

Represent input image as a vector $\mathbf{x} \in \mathbb{R}^{784}$
Learn a classifier $f(\mathbf{x})$ such that,

$$f : \mathbf{x} \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Predict the digit
Multi-class classifier

Parameter Counting

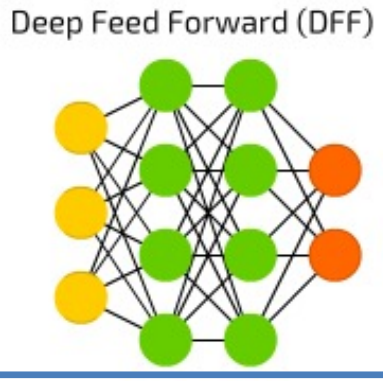
Review FFNN

- Feed-Forward Neural Networks are common neural networks architectures
  - Fully connected networks are called Multi-Layer Perceptron
  - Usually use 1 or 2 hidden layers
- Input, output, and hidden layers
  - Linear matrix operations followed by non-linear activations at every layer
- Activations:
  - ReLU, tanh, etc., for hidden layers
  - Sigmoid (binary classification) and softmax (for multi-class classification) at last layer
- Forward propagation: process of evaluating input through the network

Neural Network Architectures

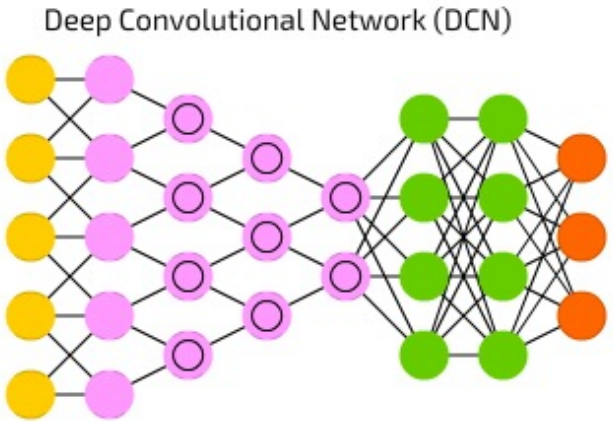
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



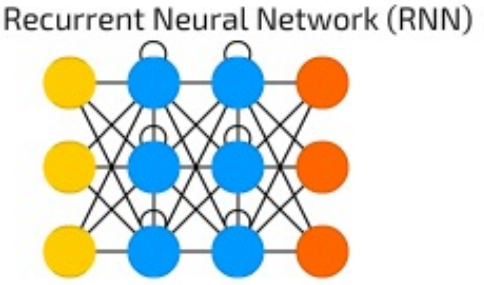
Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations



Recurrent Networks

- Keep hidden state
- Have cycles in computational graph

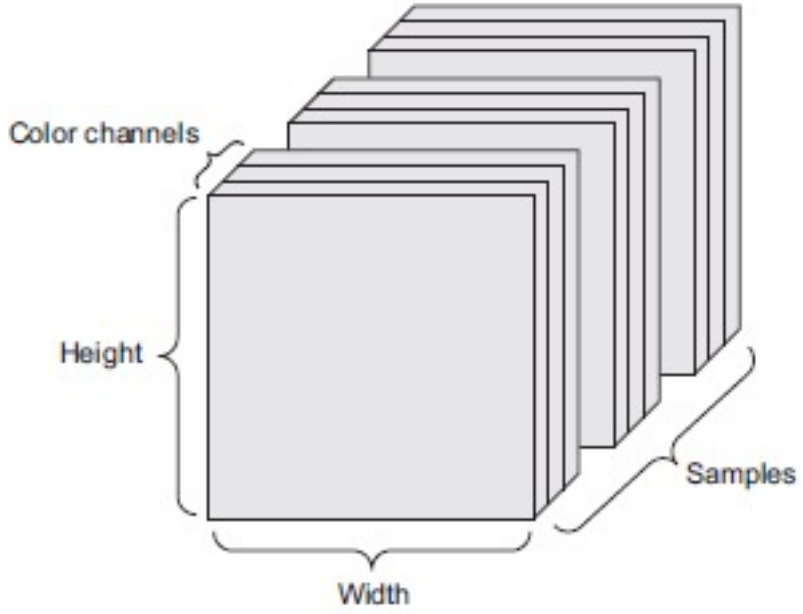


Convolutional Nets

- Neurons are connected from layer to the next
  - Invented by [LeCun 89]
- Applicable to data with natural grid topology
  - Time series
  - Images
- Use convolutions on at least one layer
  - Convolution is a linear operation that uses local information
  - Also use pooling operation
  - Used for dimensionality reduction and learning hierarchical feature representations

Image Representation

- Image is 3D “tensor”: height, width, color channel (RGB)
- Black-and-white images are 2D matrices: height, width
  - Each value is pixel intensity

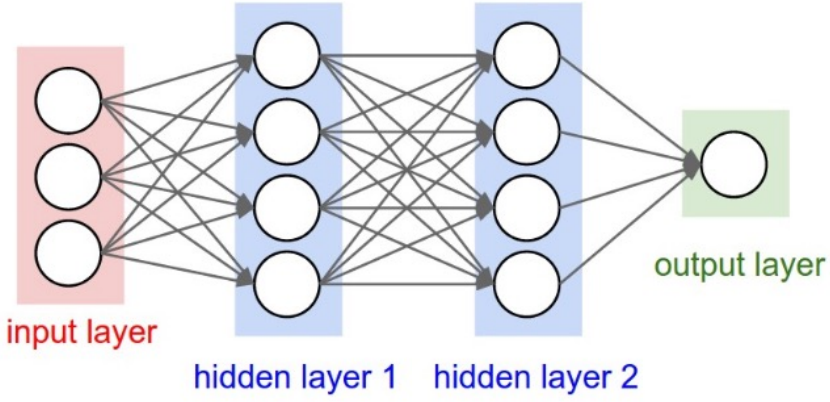


Computer vision principles

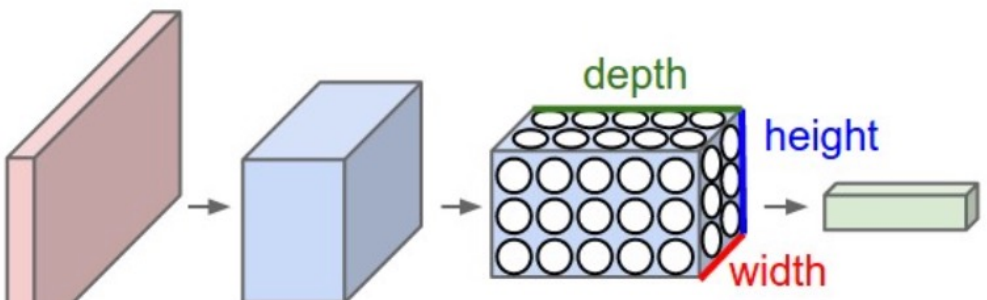
- Task: image classification (object identification)
- Translation invariance
  - Classification should work if object appears in different locations in the image => All image regions are treated the same
- Locality
  - Focus on local regions for object detection => computation should be local
- Mathematical operation: Convolution

Convolutional Neural Networks

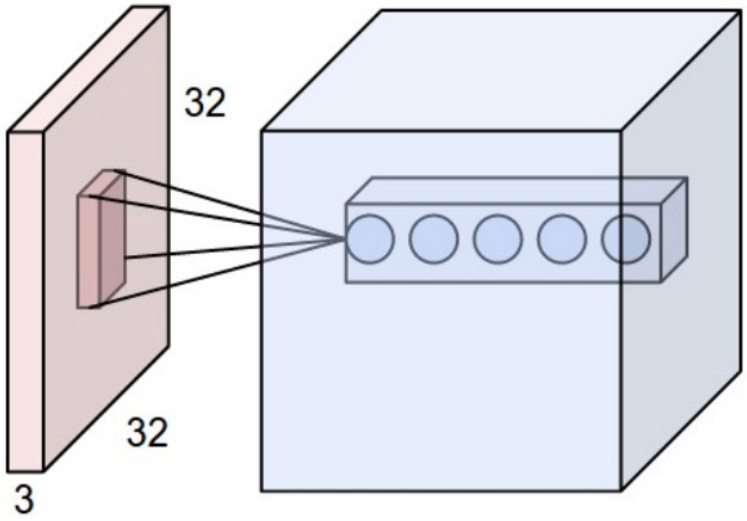
Feed-forward network



Convolutional network

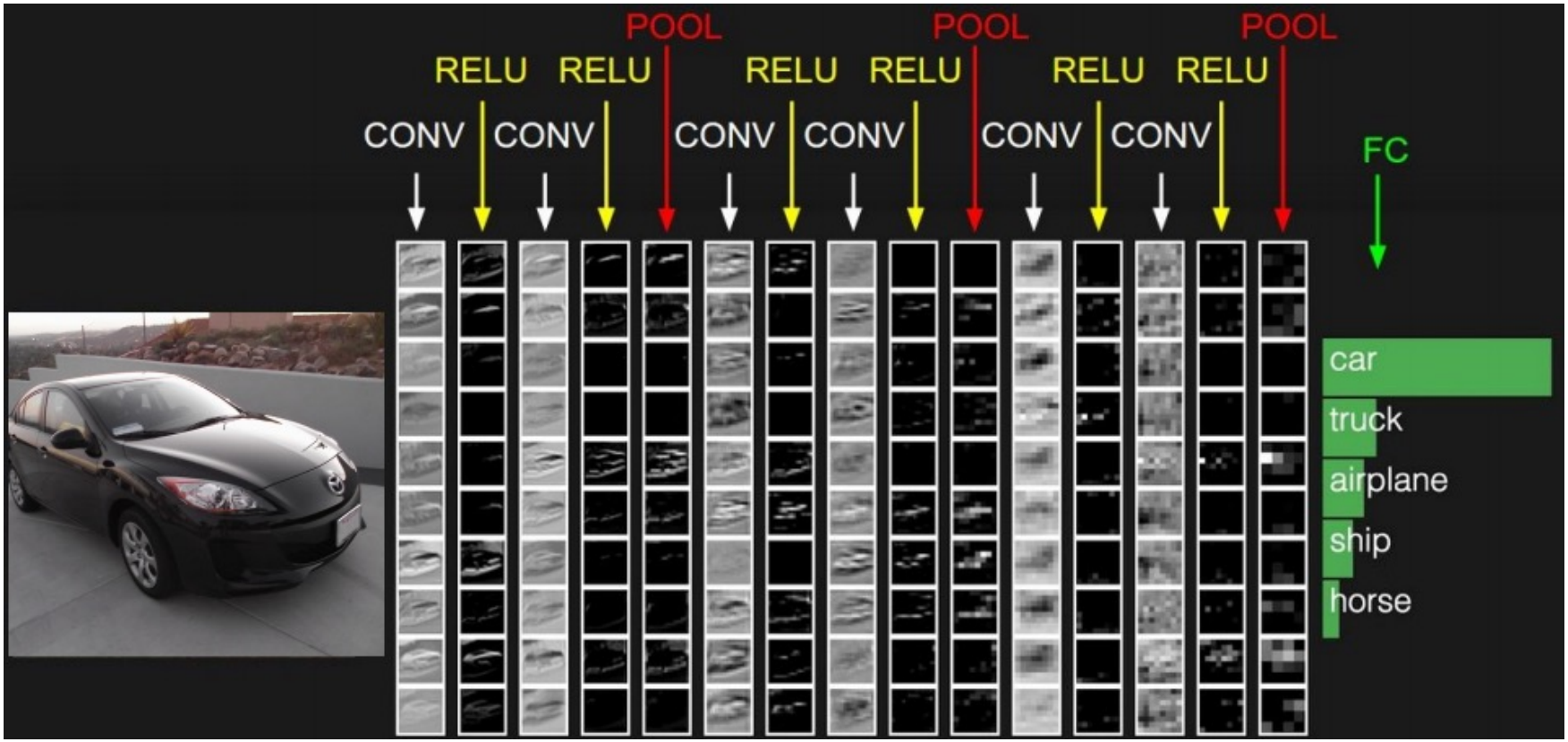


Filter



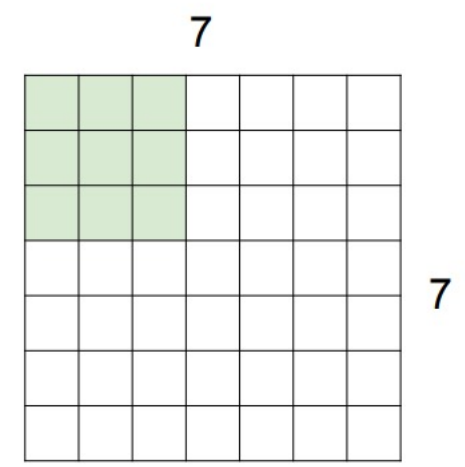
https://www.youtube.com/watch?v=f0t-OCG79-U

Convolutional Nets

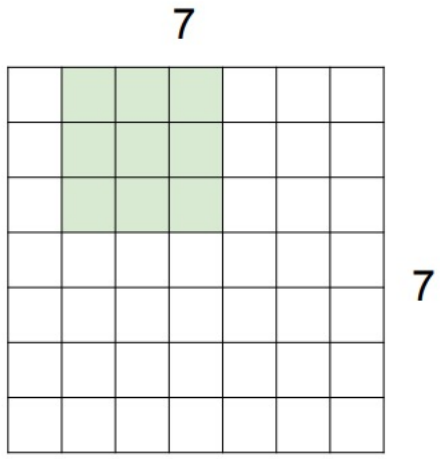


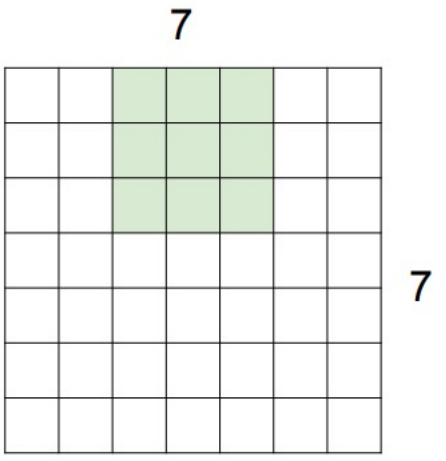
Convolutions

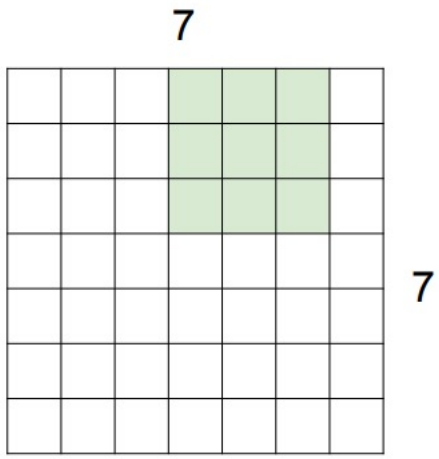
A closer look at spatial dimensions:

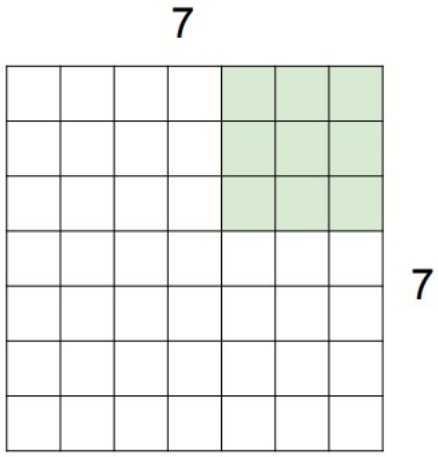


7x7 input (spatially)
assume 3x3 filter

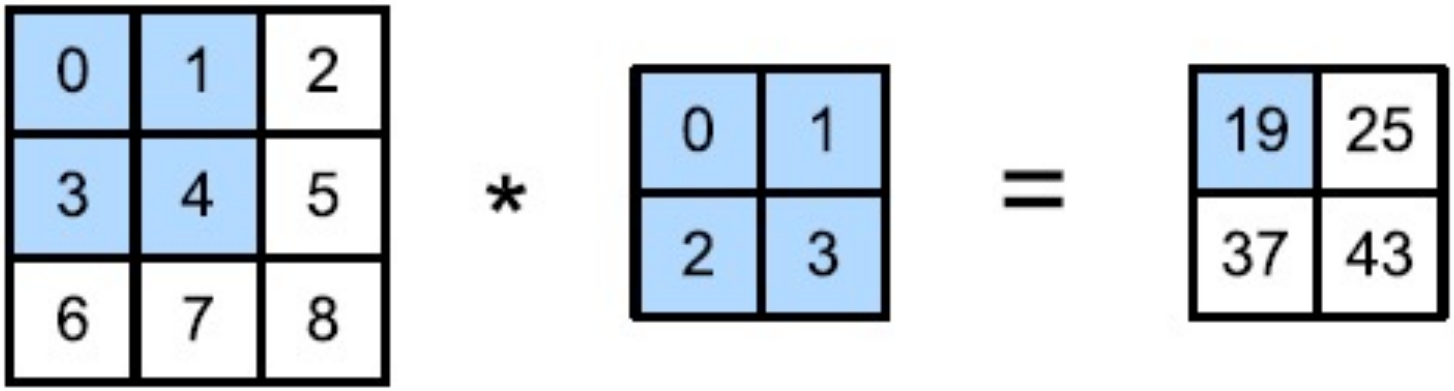








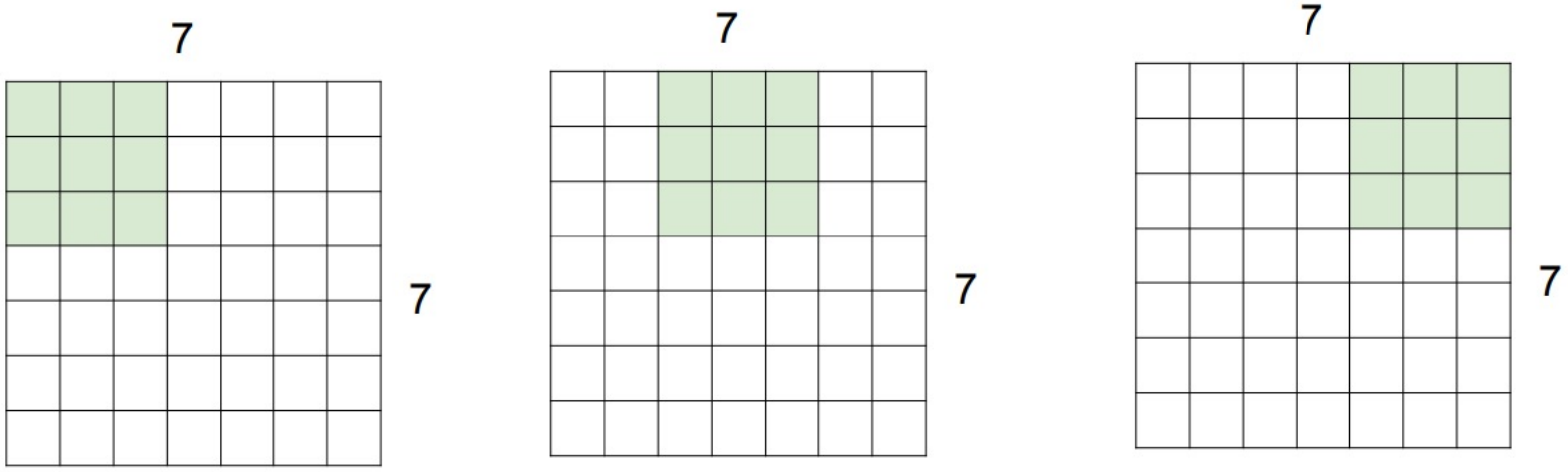
Example



Input Filter Output

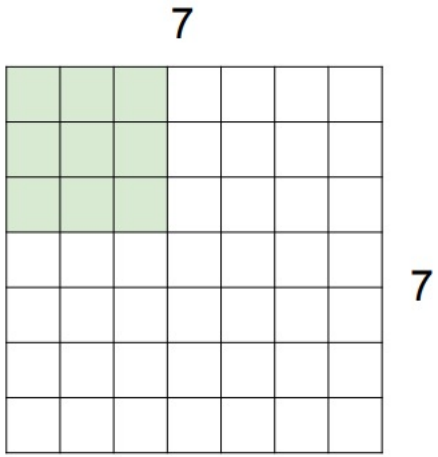
Convolutions with stride

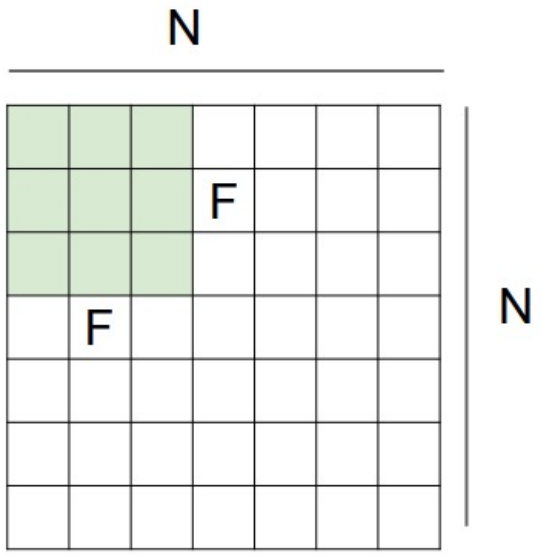
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**



Convolutions with stride

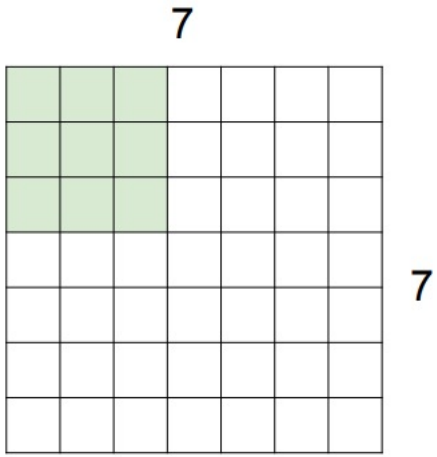
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**



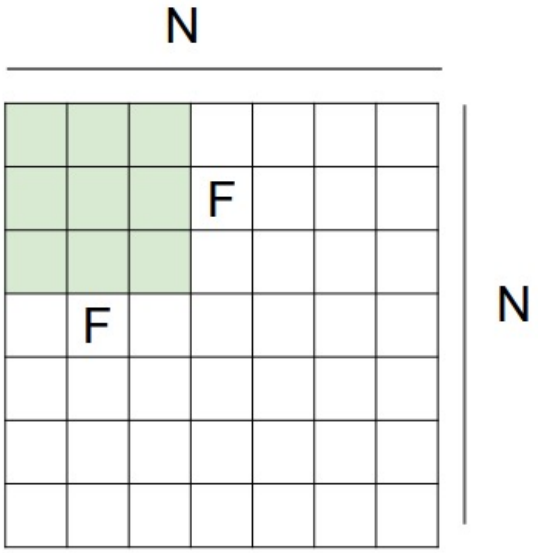


Convolutions with stride

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



doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

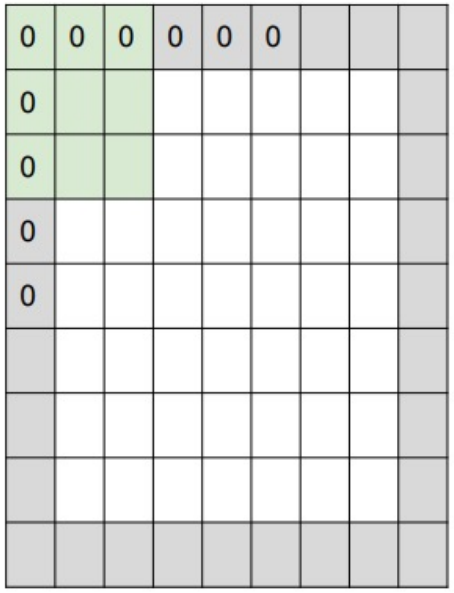


Output size:
(N - F) / stride + 1

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 :\

Padding

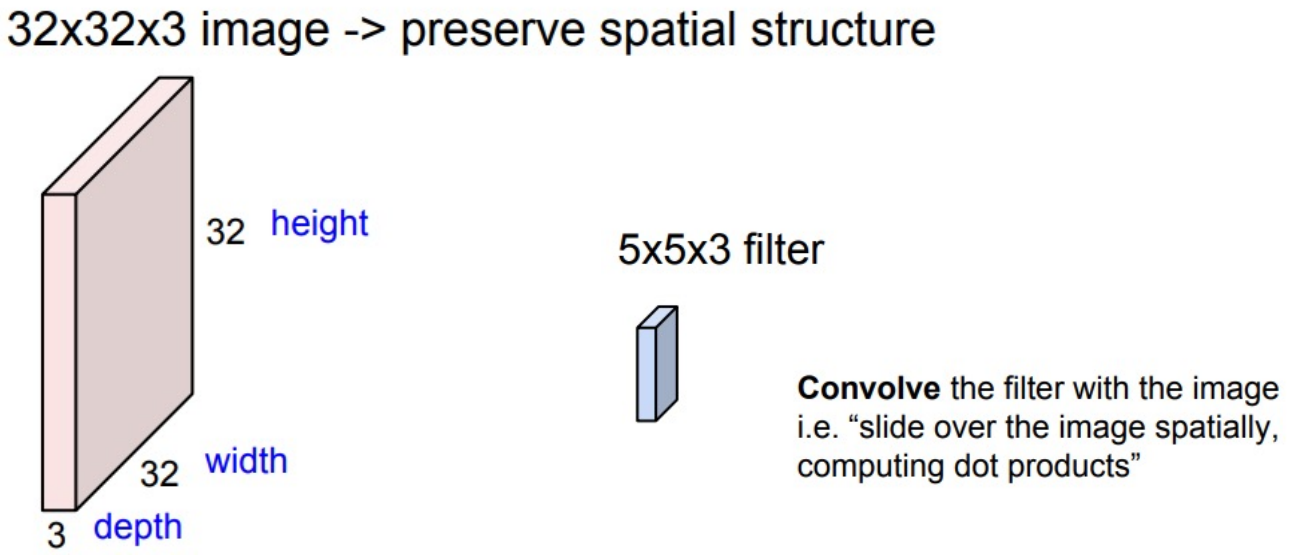
In practice: Common to zero pad the border



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

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

Convolution Layer



- Depth of filter always depth of input
- Computation is based only on local information

Convolution layer: Takeaways

- Convolution is a linear operation
  - Reduces parameter space of Feed-Forward Neural Network considerably
  - Capture locality of pixels in images
  - Smaller filters need less parameters
  - Multiple filters in each layer (computation can be done in parallel)
- Convolutions are followed by activation functions
  - Typically ReLU