

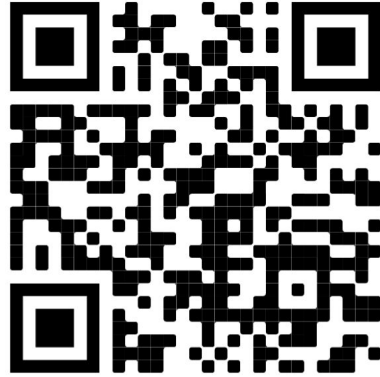


BOSTON UNIVERSITY
MACHINE INTELLIGENCE
COMMUNITY

Common ML Models

Darcy, Kirsten
10/14/2021

Attendance

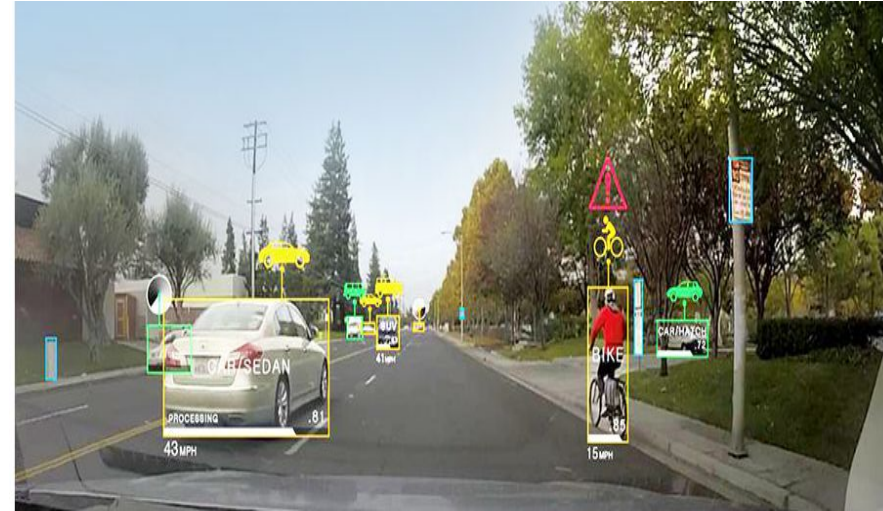


<https://forms.gle/1YDi83J3rvaWUanM8>

Applications of Deep Learning

1. Cool things using deep learning

- a. Computer Vision
 - i. Tesla recognizing items on a street
- b. Text generation
 - i. An algorithm was trained to create a similar Shakespeare piece
- c. Image recognition
 - i. Classifying what a certain picture contains
 - ii. Facebook photo tagging
- d. Many more...

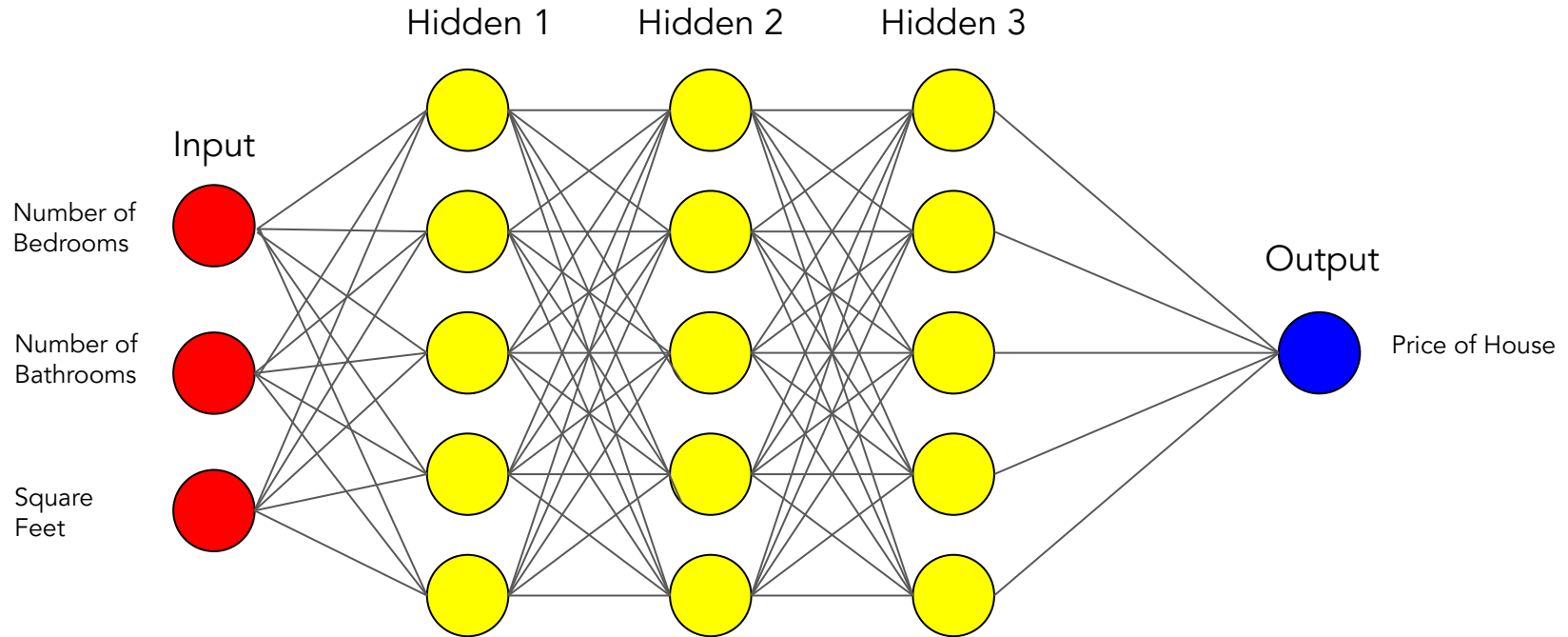




Deep Learning Process

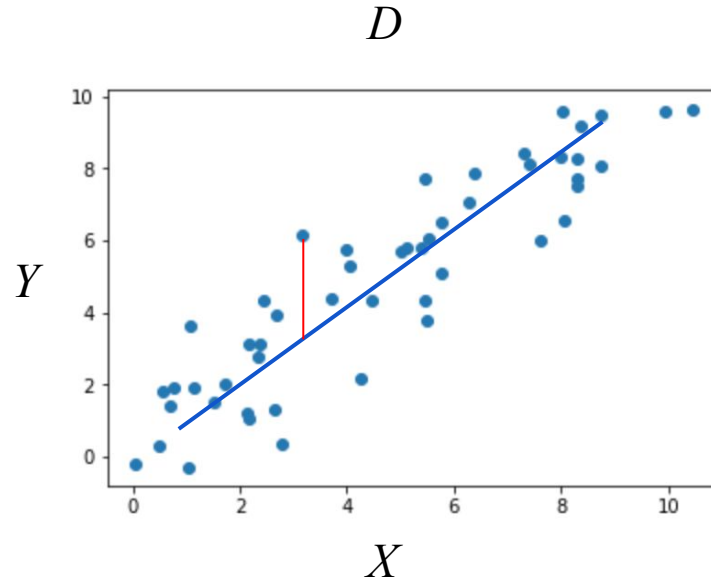
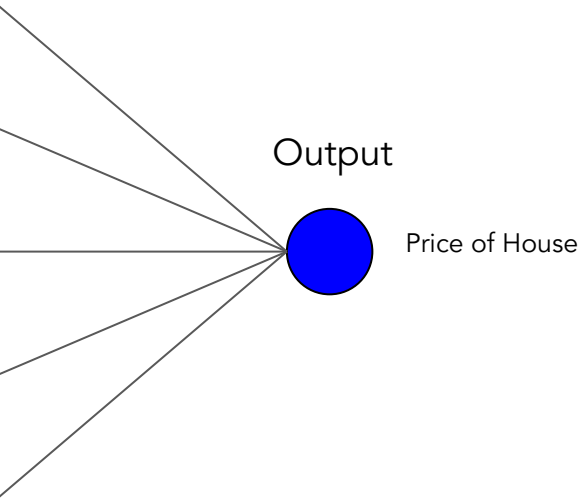
Forward propagation

Push example through the network to get a predicted output



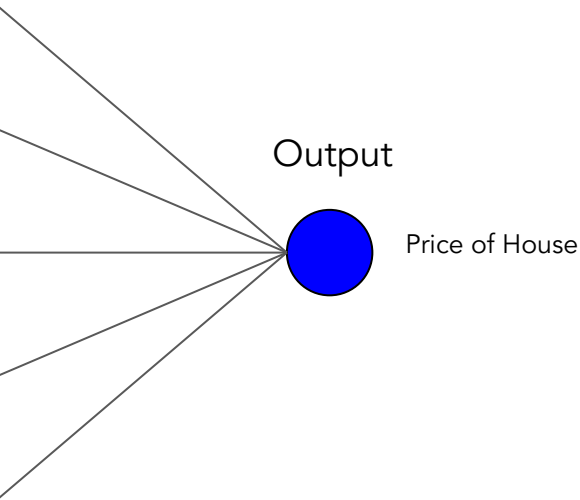
Compute the cost

Calculate difference between predicted output and actual data



Compute the cost

Calculate difference between predicted output and actual data

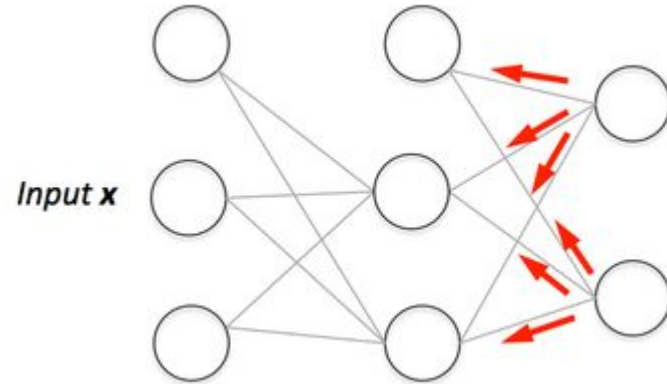
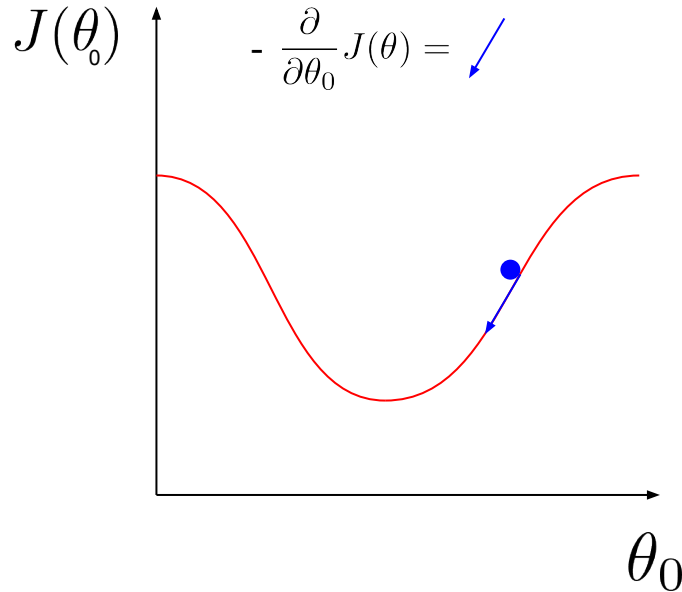


$$J(\theta) = \frac{1}{2m} \sum_i^m (y_i - \hat{y}_i)^2$$

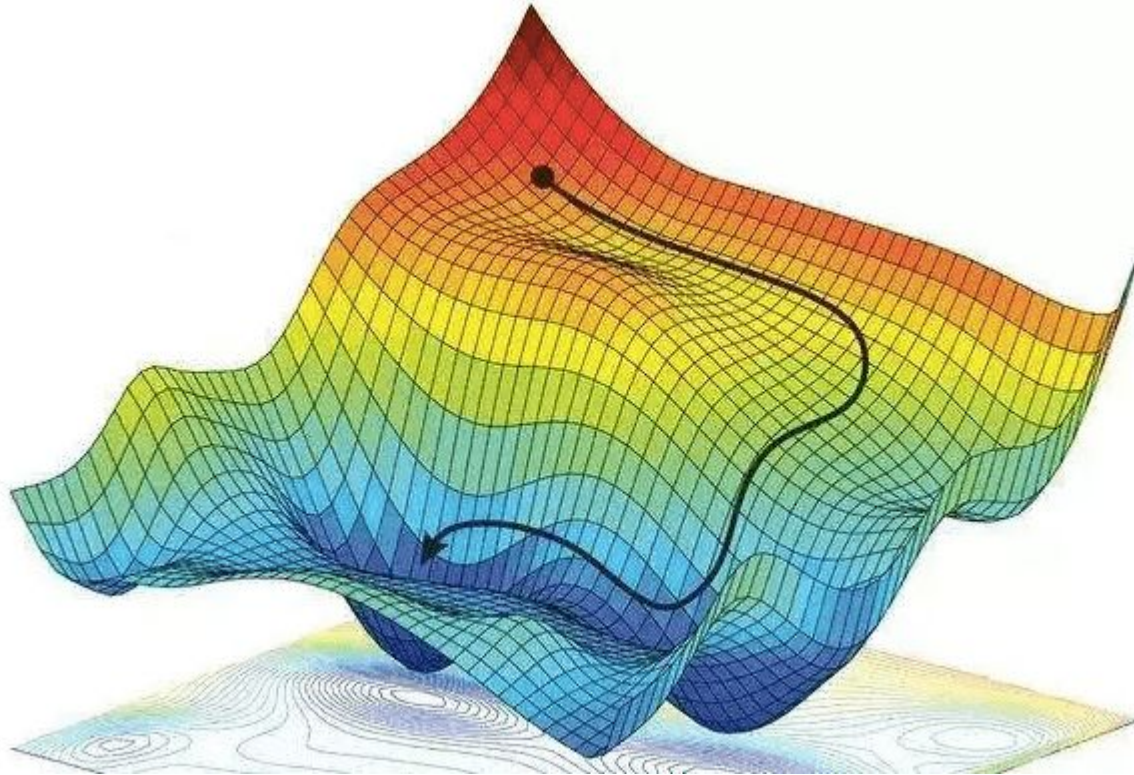
Where i is the i th training example and m is the number of training examples

Backward propagation - "Update"

Push back the derivative of the error and apply to each weight, such that next time it will result in a lower error



Cost function for gradient descent



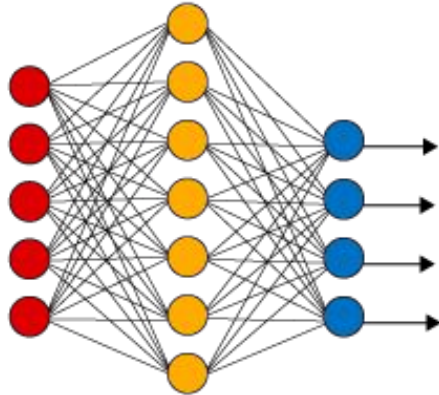
A decorative graphic on the left side of the slide, consisting of several curved lines in shades of orange, pink, and purple. These lines connect circular nodes of corresponding colors, forming a stylized representation of a neural network or a complex system. The nodes are of varying sizes, and the lines are smooth and flowing.

Deep Neural Networks

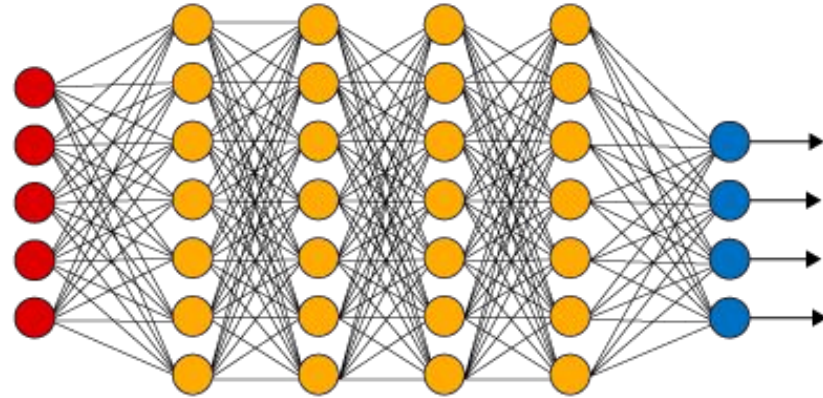
Deep Neural Networks

- Just a neural network with more layers

Simple Neural Network



Deep Learning Neural Network

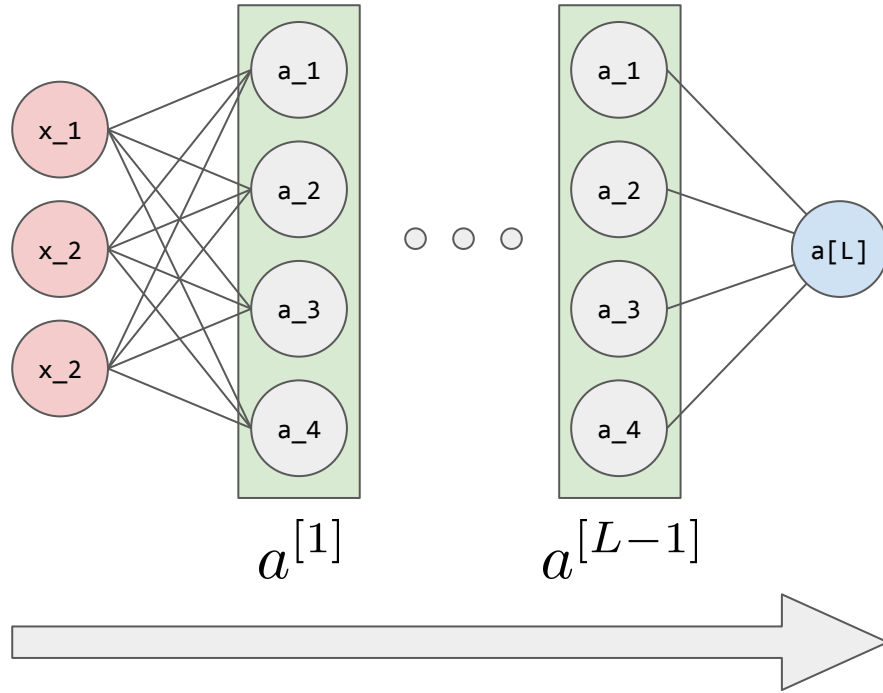


● Input Layer

● Hidden Layer

● Output Layer

Forward Propagation



Repeat $L-1$ times

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}, \quad a^{[l]} = g^{[l]}(z^{[l]})$$

The equation is visualized using 3D tensors. On the left is the output tensor $z^{[l]}$. This is equal to the product of the weight tensor $W^{[l]}$ and the input tensor $a^{[l-1]}$, plus the bias tensor $b^{[l]}$. The weight tensor $W^{[l]}$ is shown as a 3D grid with dimensions $n_h^{[l-1]} \times n_h^{[l-1]} \times m$. The input tensor $a^{[l-1]}$ has dimension m . The bias tensor $b^{[l]}$ has dimension $n_h^{[l]}$.

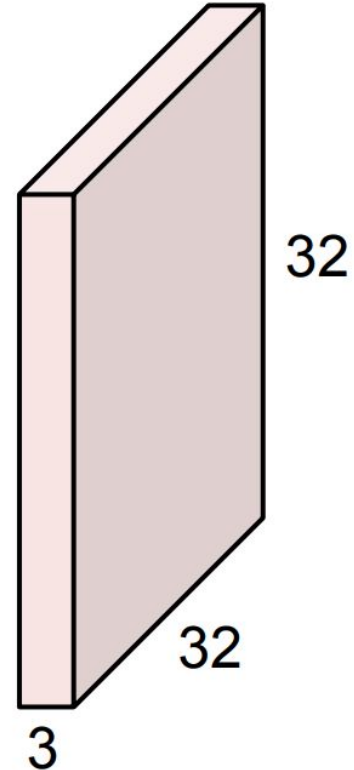


Convolutional Neural Networks (CNNs)

Image Data

- Images are commonly represented in code as a 3D array of pixels. Here, we notice 3 represents RGB values
- In vanilla neural networks, we would simply flatten this 3D array into a 3072 length vector. However, by doing this, we lose spatial correlation between pixels close to other pixels

32x32x3 image



Convolutional Operation

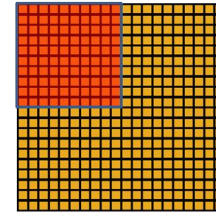
1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Pooling Layers



Convolved
feature



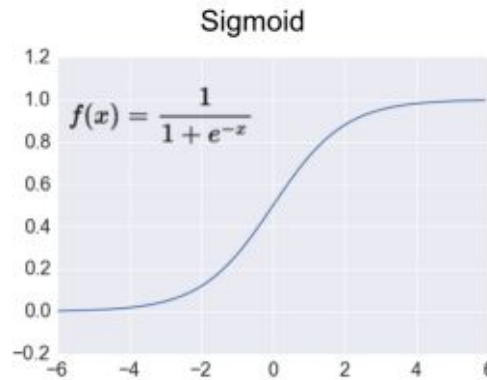
Pooled
feature

- Limitation of output of Convolutional Layers:
 - Record the precise position of features in the input
 - Small movements in the position of the feature in the input image will result in a different feature map
- Solution: Pooling Layers
 - Lower resolution version of input is created with large and important structure elements preserved
 - Reduces the computational cost by reducing the number of parameters to learn

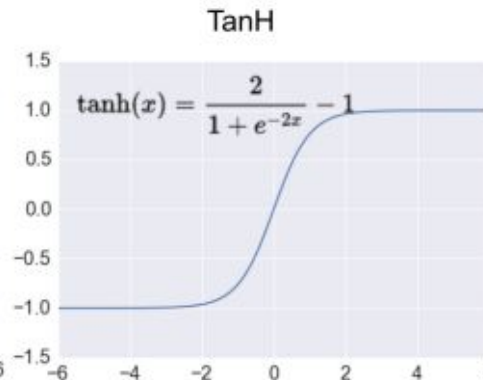
Activation Functions

Activation Functions model nonlinear data by taking inputs and comparing them to a threshold. This allows us to model non-linear data.

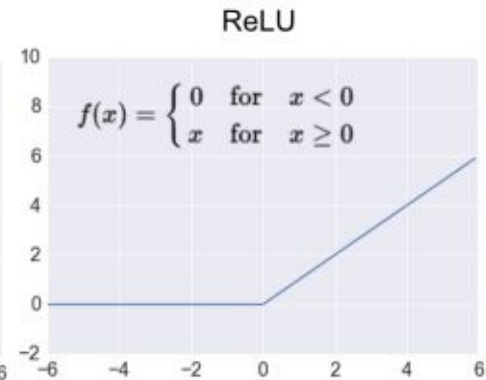
Sigmoid: output is
between 0,1



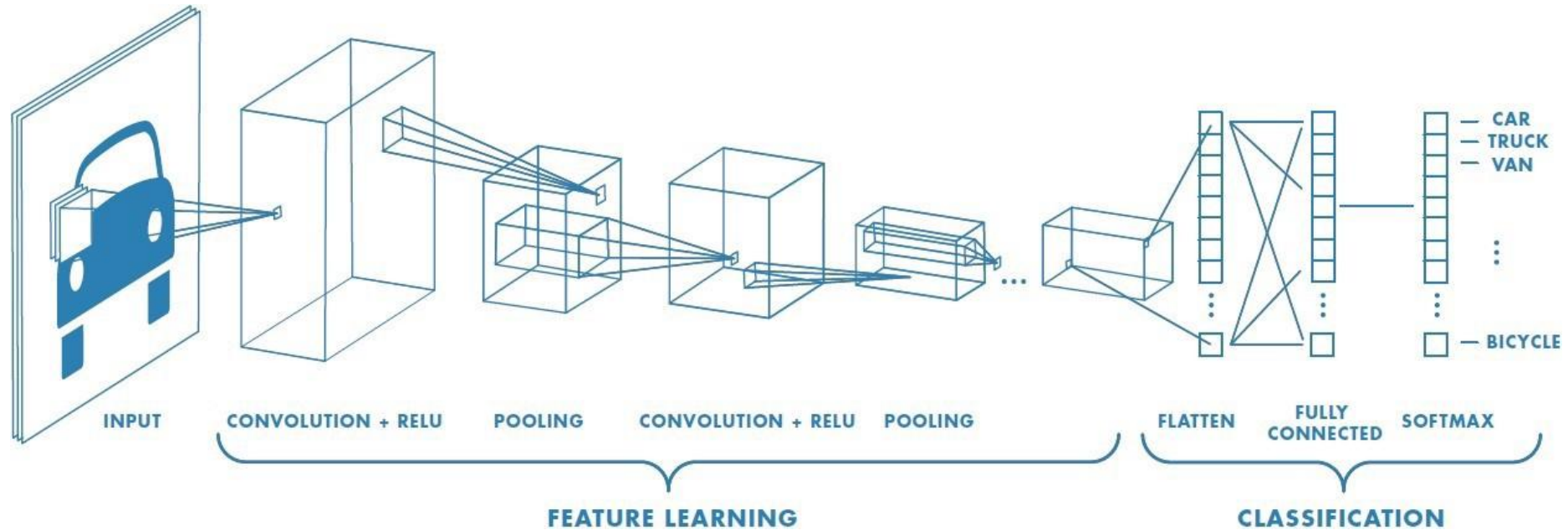
Tanh: output is
between -1,1



ReLu: output is
positive real numbers



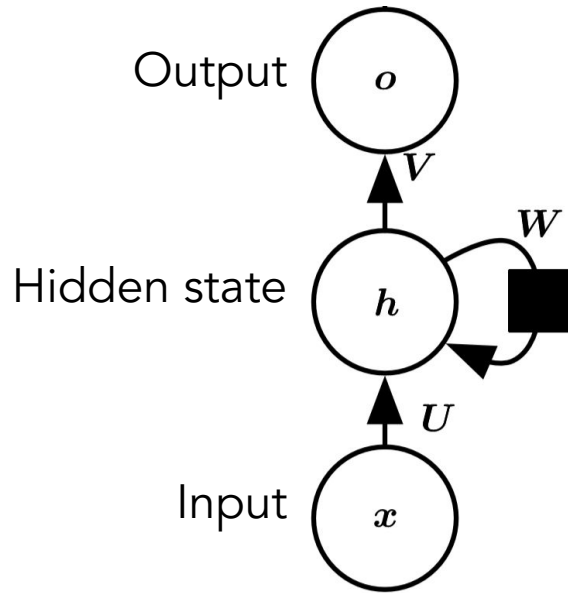
Convolutions



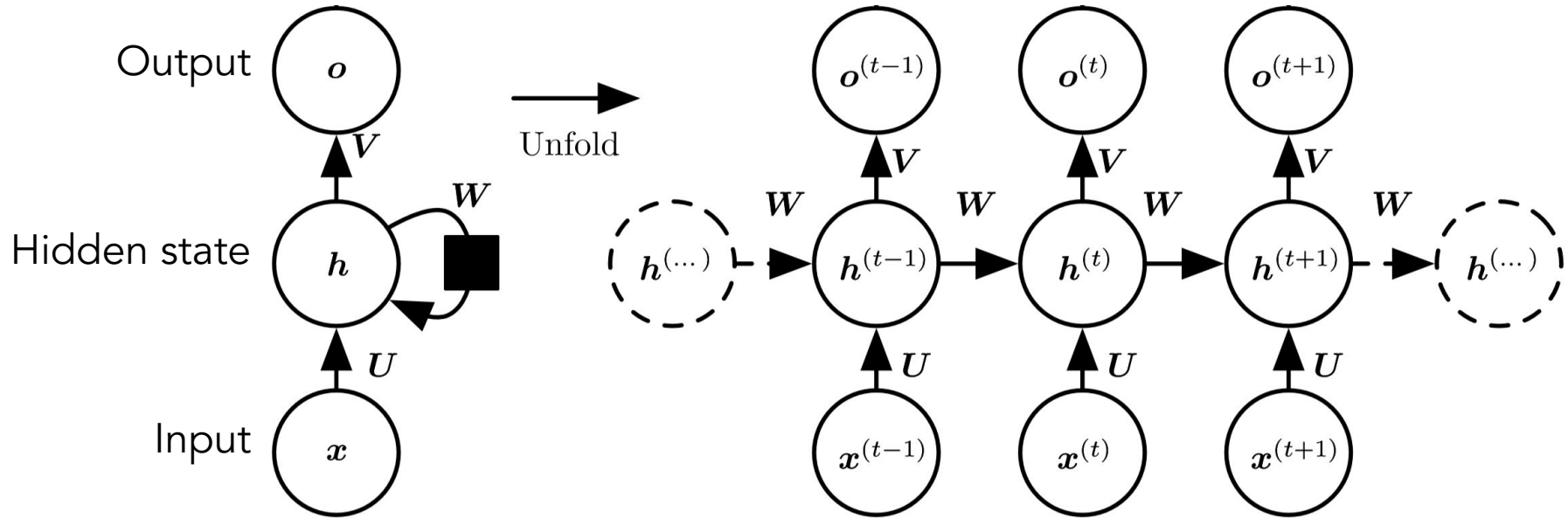


Recurrent Neural Networks (RNNs)

RNN Cell



RNN Graph



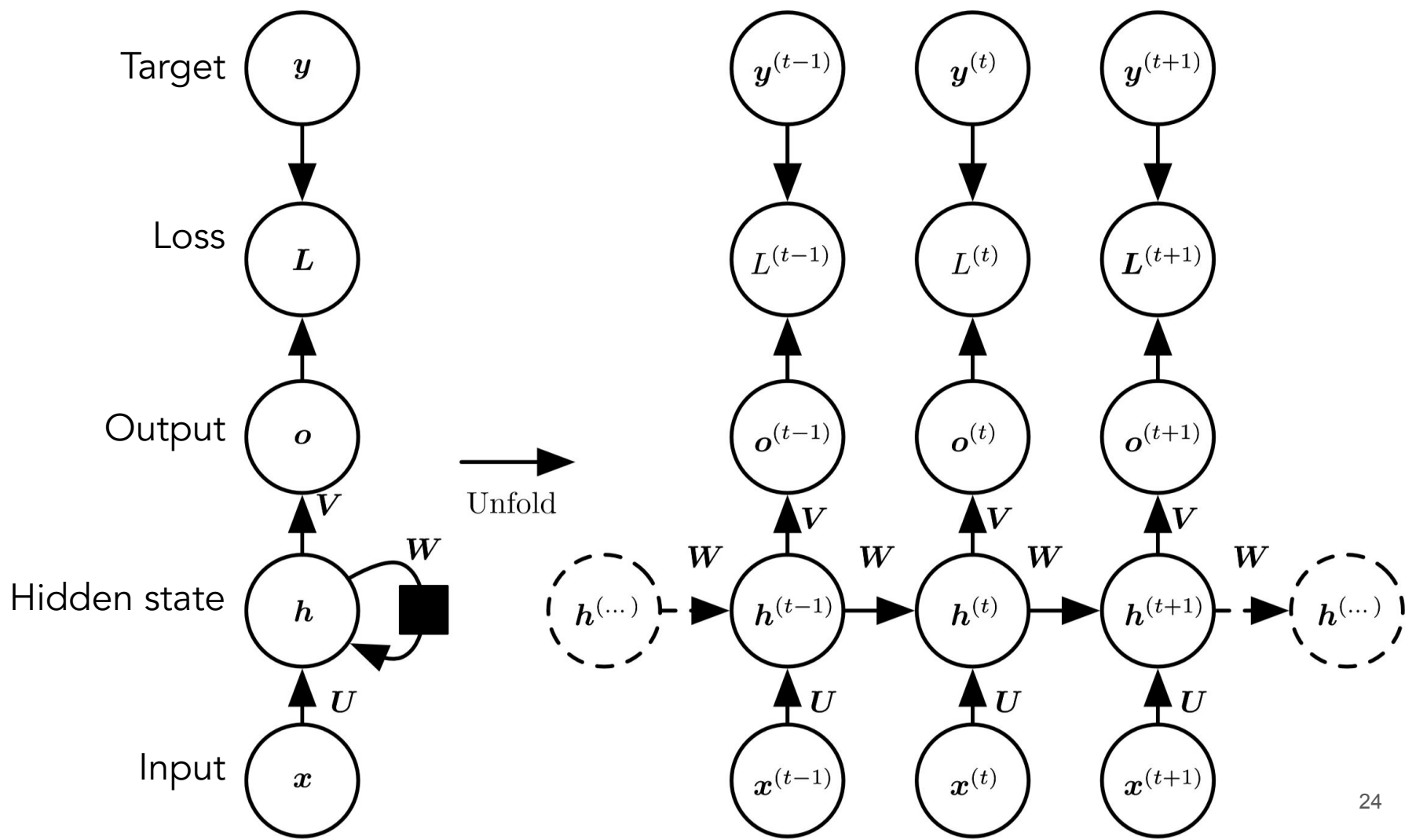
RNN Feedforward

Affine $\mathbf{a}^{(t)} = \underbrace{\mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}}_{\text{Affine function (hidden network)}},$

Hidden state $\mathbf{h}^{(t)} = \underbrace{\tanh(\mathbf{a}^{(t)})}_{\text{Activation function}},$

Output $\mathbf{o}^{(t)} = \underbrace{\mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}}_{\text{Output network}},$







PyTorch Tutorial - CNN

<https://tinyurl.com/2j27t5se>



Eboard positions available!

<https://forms.gle/aV12v3iJVMnRb1xo6>