Common ML Models

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

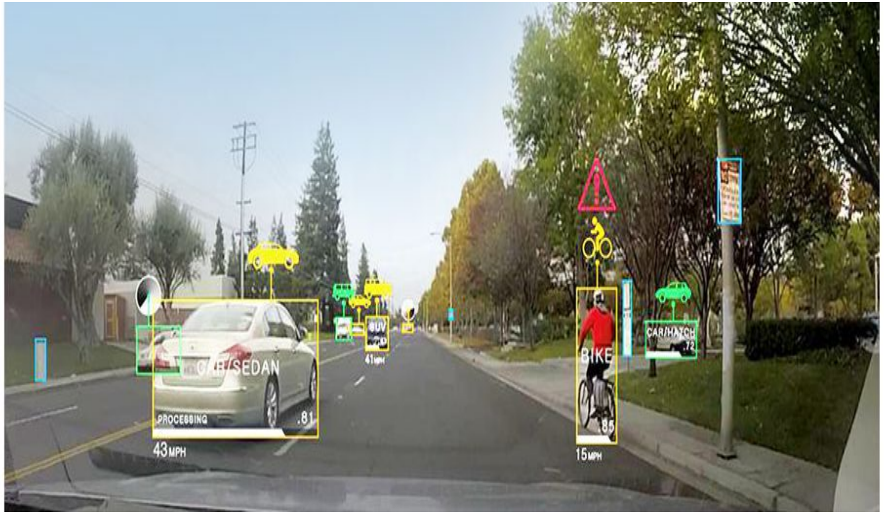
Darcy, Kirsten
10/14/2021

Attendance

https://forms.gle/1YDi83J3rvaWUanM8

Applications of Deep Learning

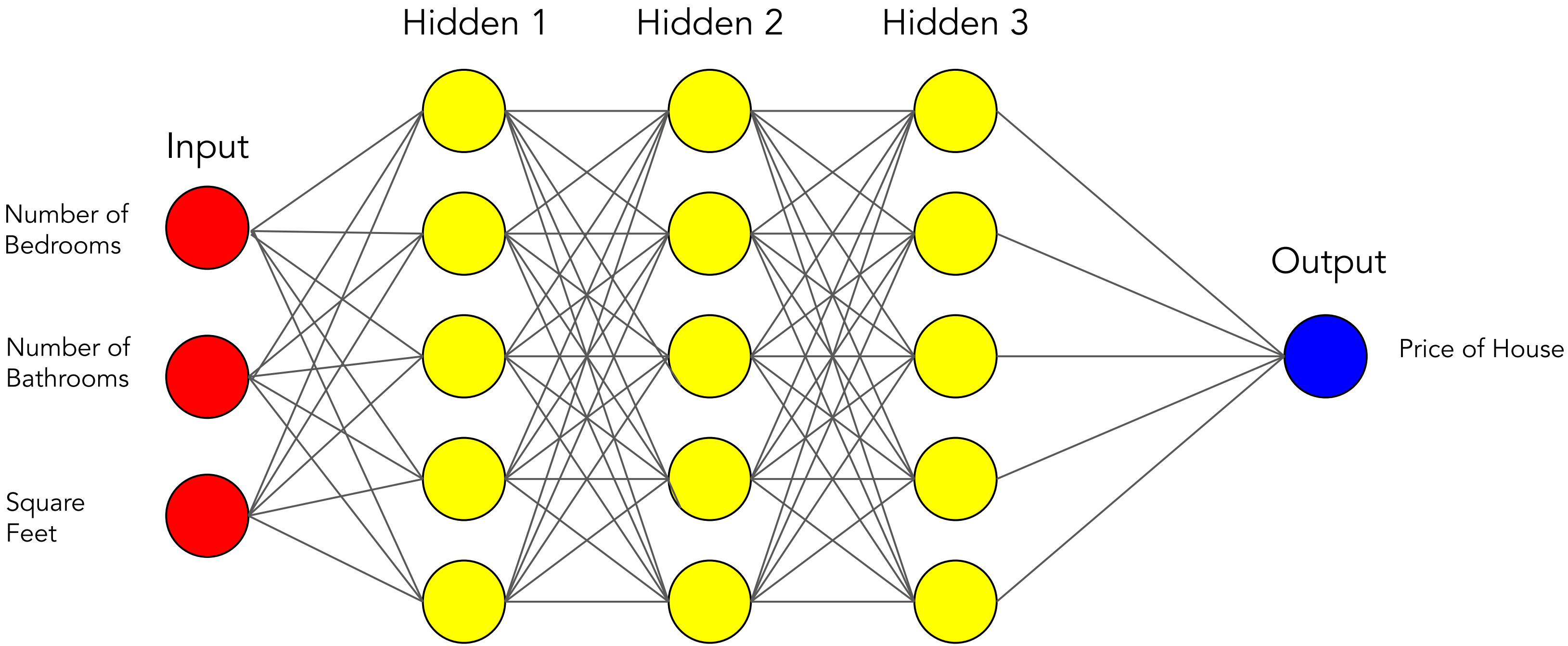
1. Cool things using deep learning
   a. Computer Vision
      i. Tesla recognizing items on a street
   b. Text generation
      i. An algorithm was trained to create a similar Shakespeare piece
   c. Image recognition
      i. Classifying what a certain picture contains
      ii. Facebook photo tagging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



Deep Learning Process

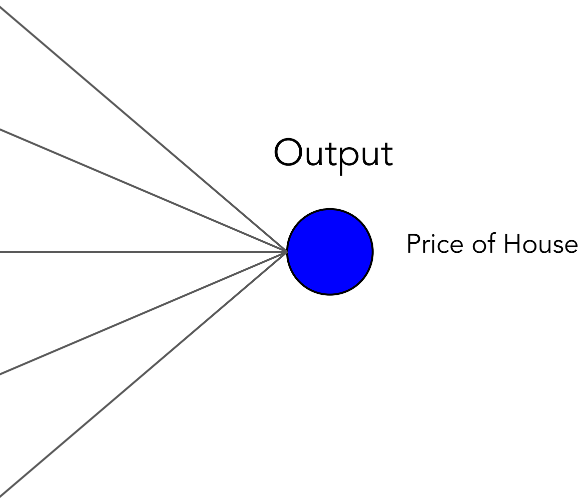
Forward propagation

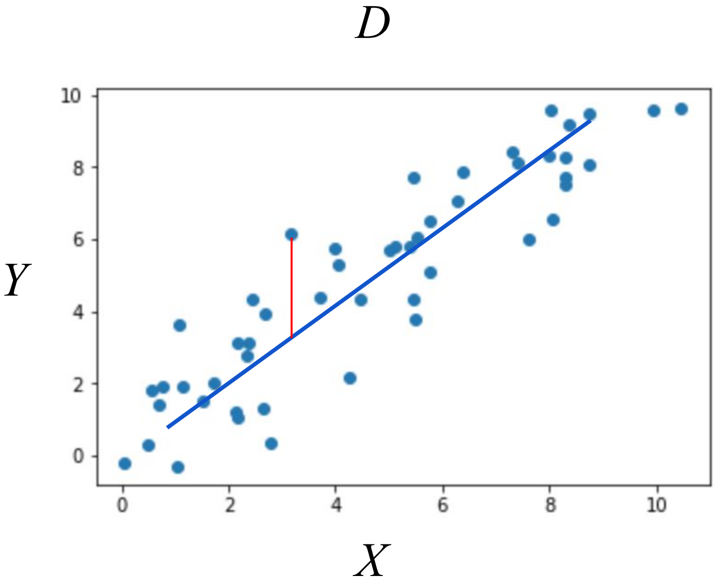
Push example through the network to get a predicted output



Compute the cost

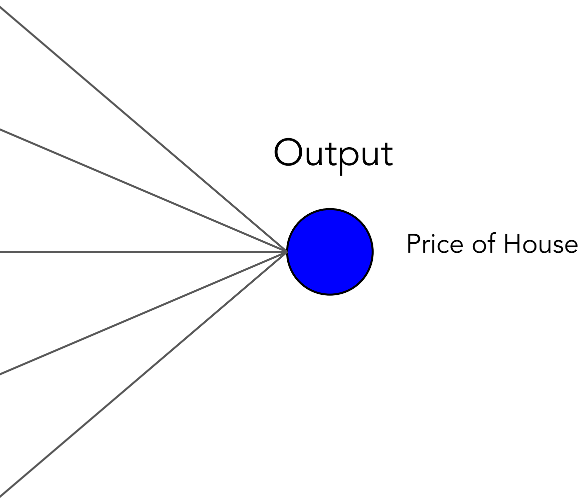
Calculate difference between predicted output and actual data





Compute the cost

Calculate difference between predicted output and actual data

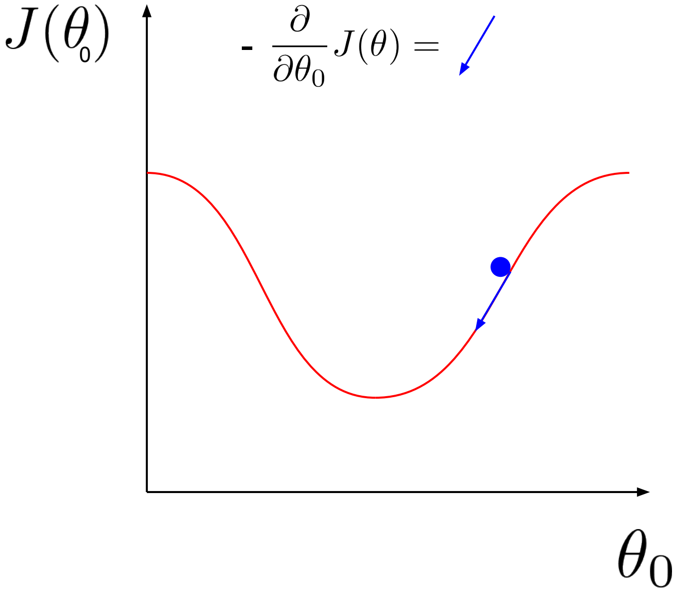


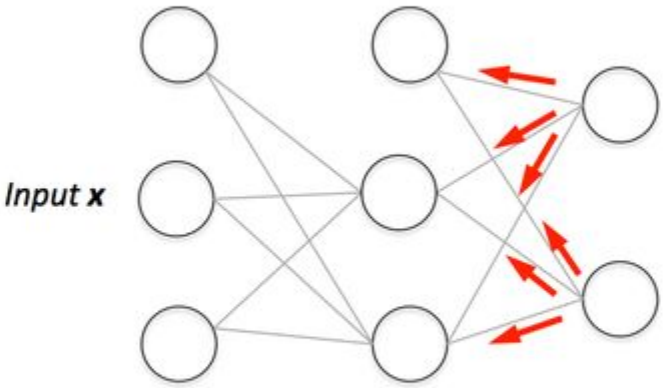
$$J(\theta) = \frac{1}{2m} \sum_{i}^{m} (y_i - \hat{y}_i)^2$$

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

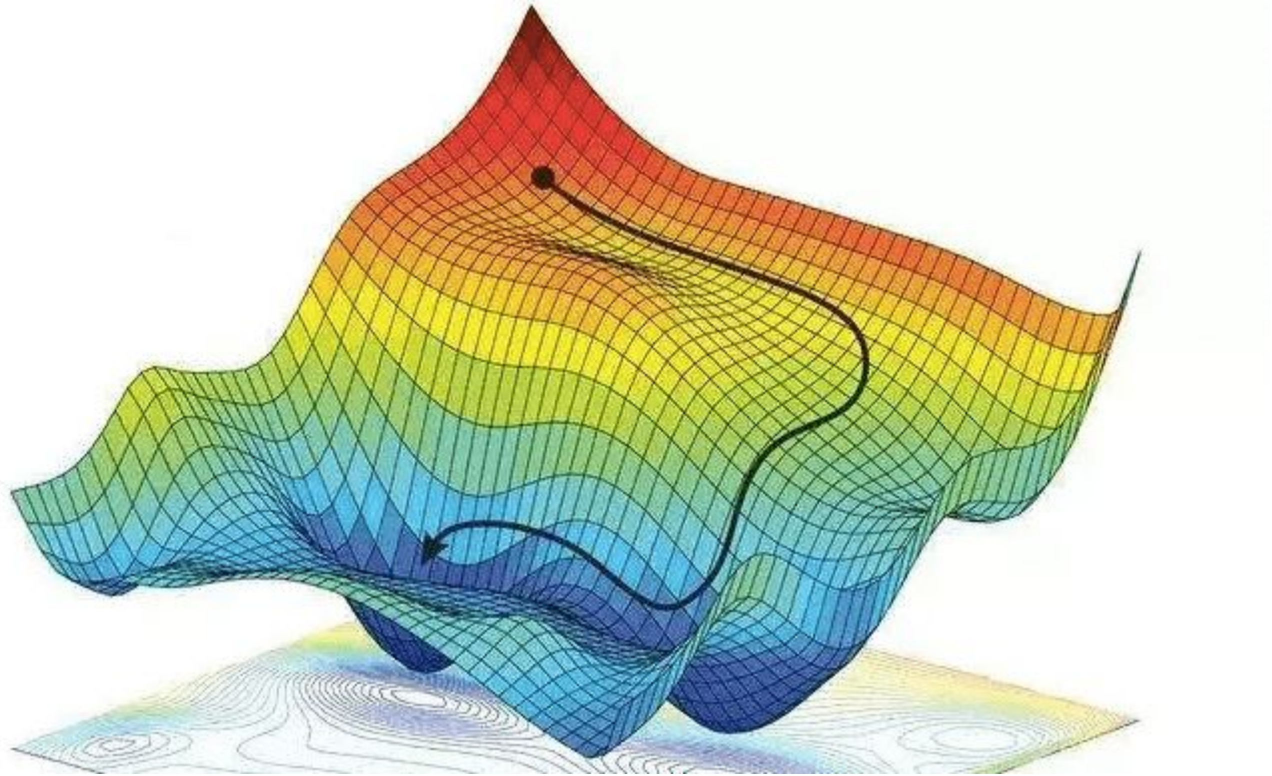
Backward propagation - "Update"

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





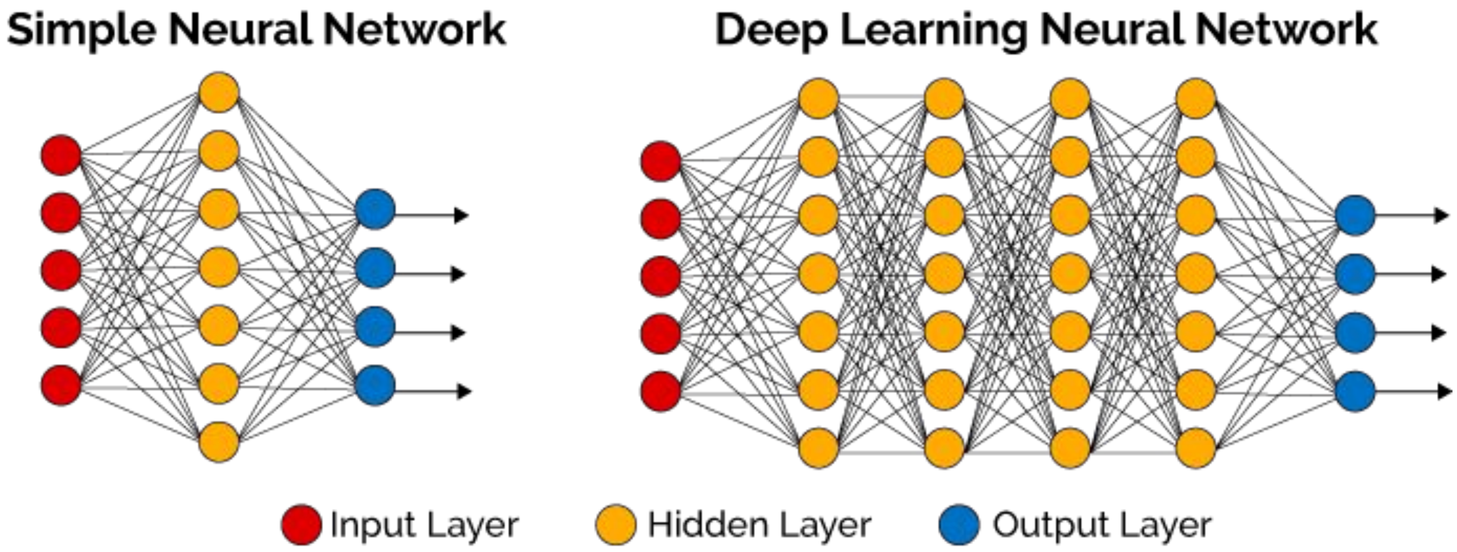
Cost function for gradient descent



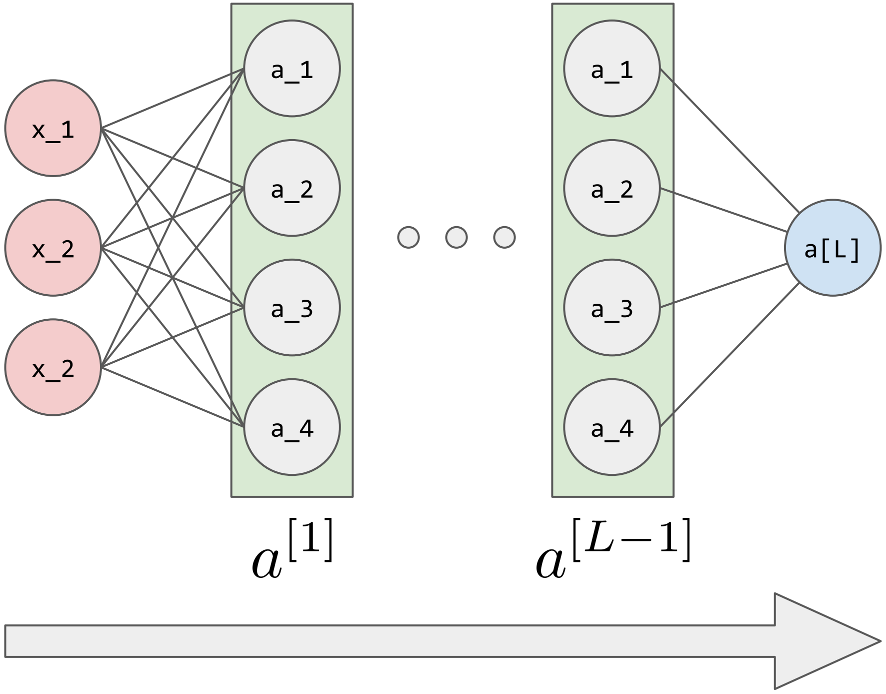
Deep Neural Networks

Deep Neural Networks

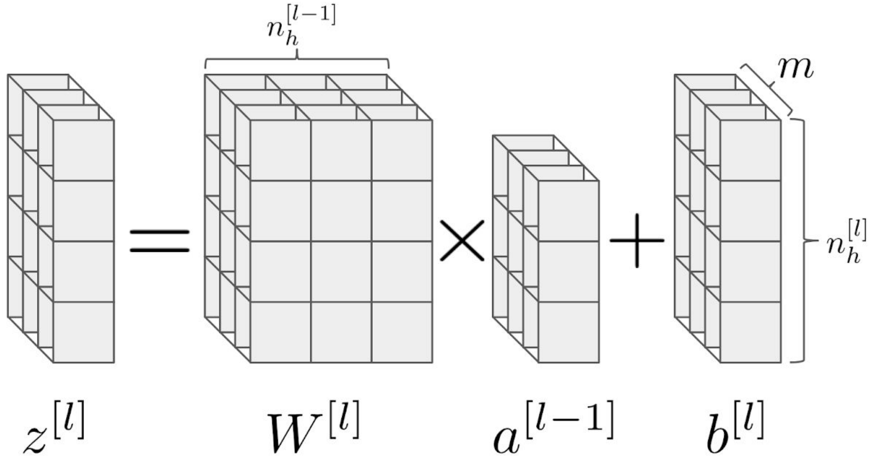
- Just a neural network with more layers



Forward Propagation



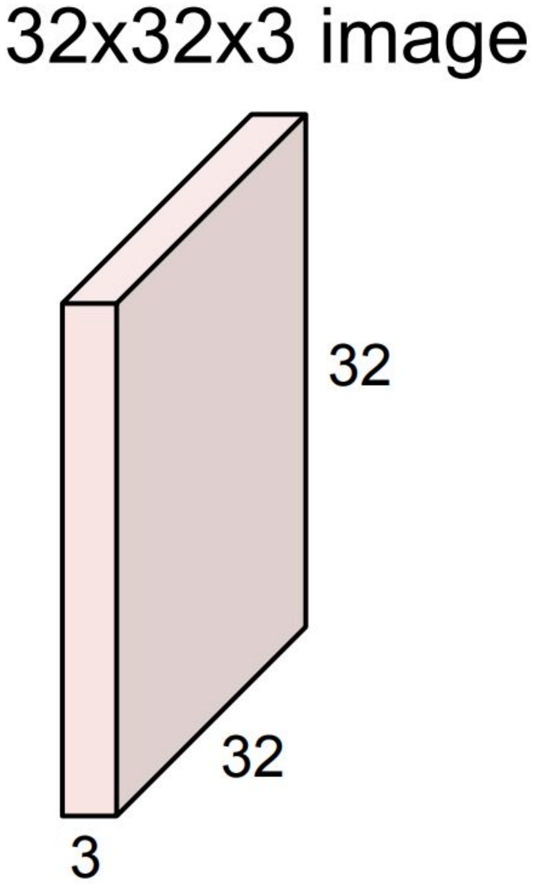
$$\underbrace{z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}, \quad a^{[l]} = g^{[l]}(z^{[l]})}_{\text{Repeat L-1 times}}$$



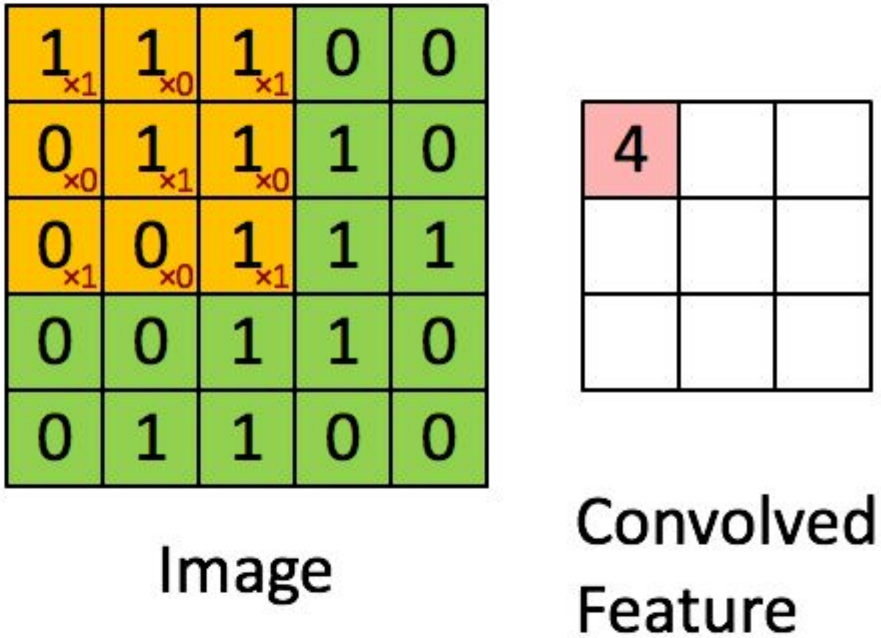
Convolutional Neural Networks (CNNs)

Image Data

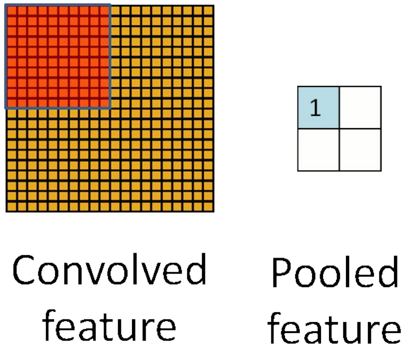
- Images are commonly represented in code as a 3D array of pixels. Here, we notice 3 represents RGB values
- In vanilla neural networks, we would simply flatten this 3D array into a 3072 length vector. However, by doing this, we lose spatial correlation between pixels close to other pixels



Convolutional Operation



Pooling Layers



- 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

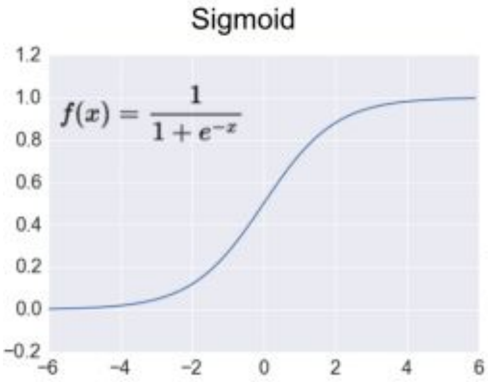
Activation Functions

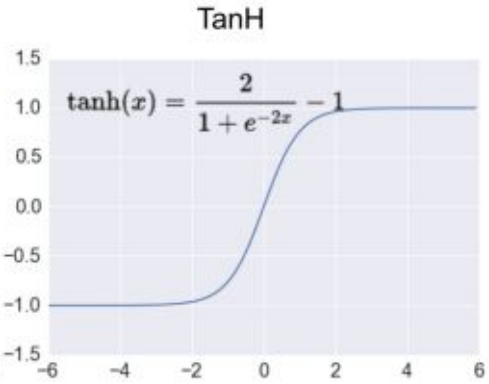
Activation Functions model nonlinear data by taking inputs and comparing them to a threshold. This allows us to model non-linear data.

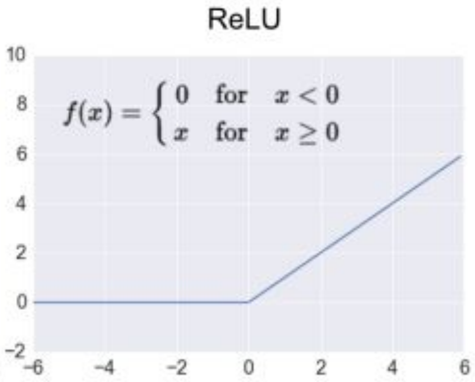
Sigmoid: output is between 0,1

Tanh: output is between -1,1

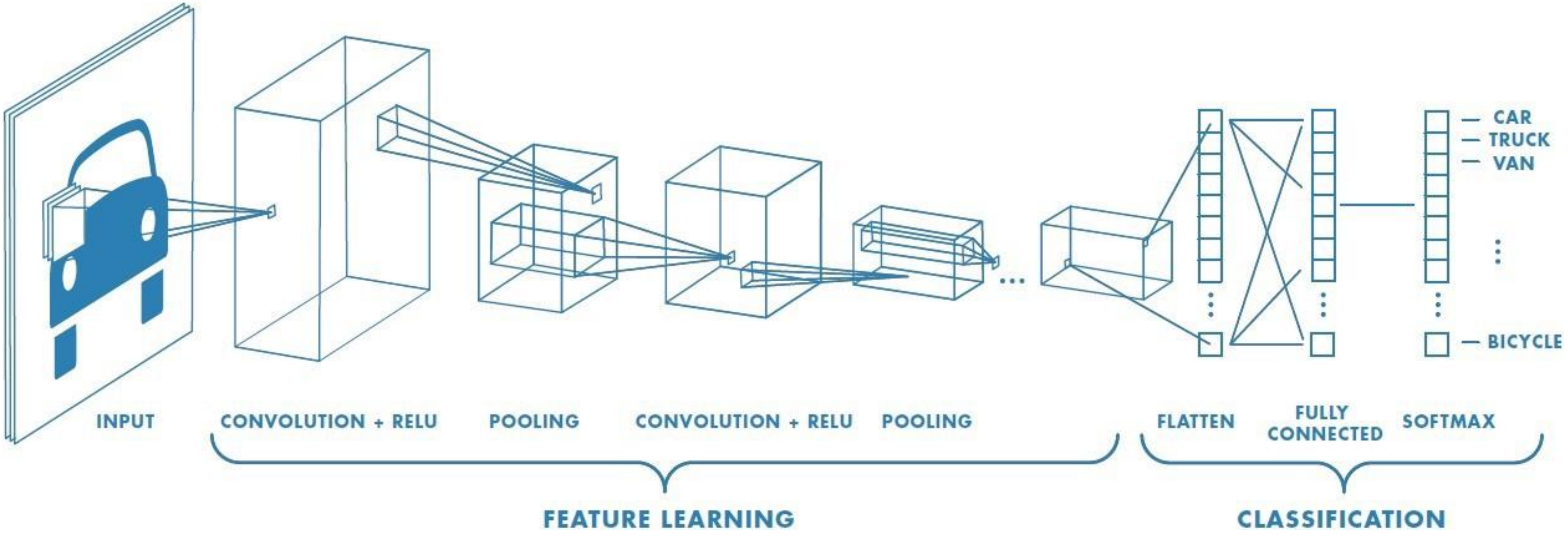
ReLu: output is positive real numbers



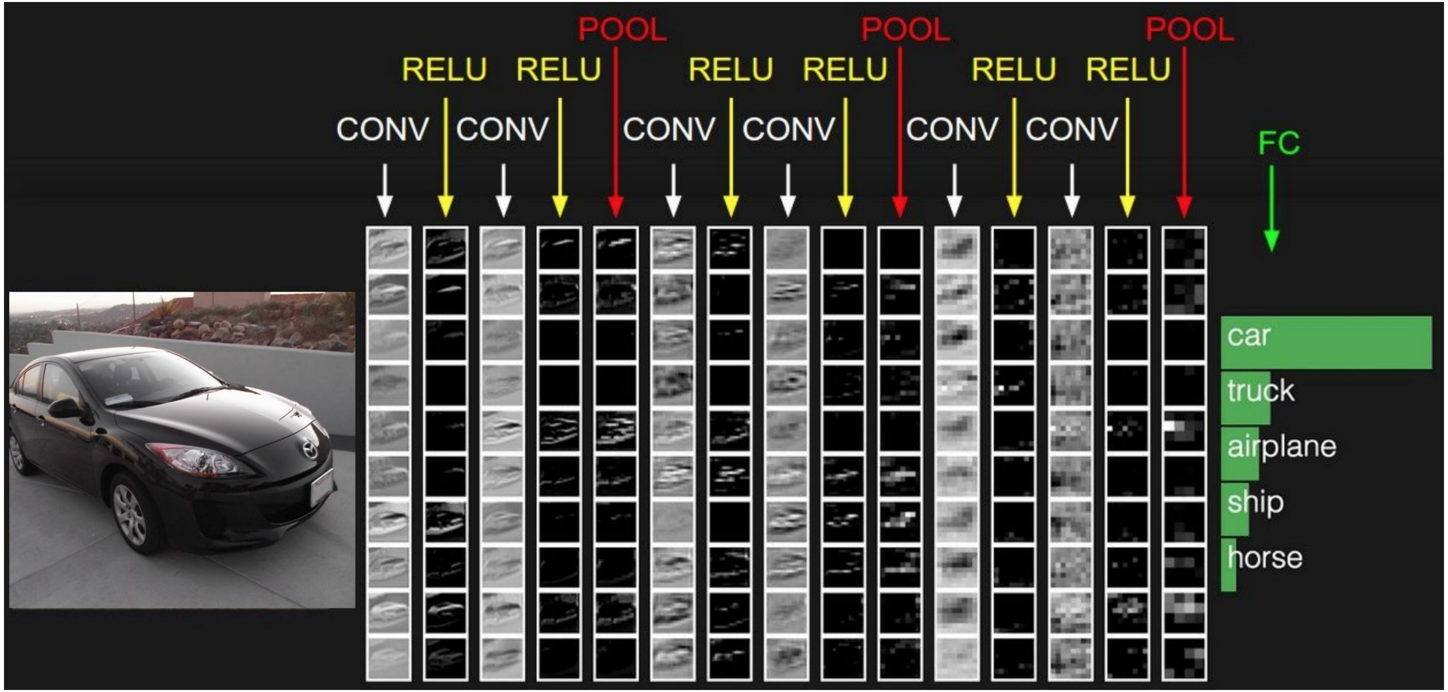




Convolutions

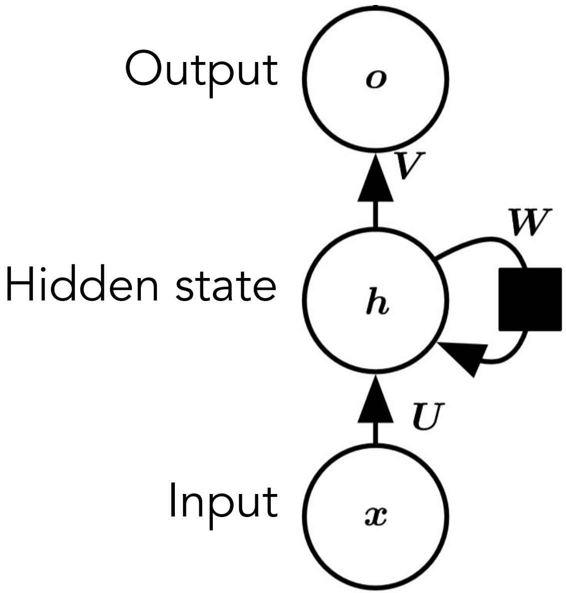


So what does our network look like?

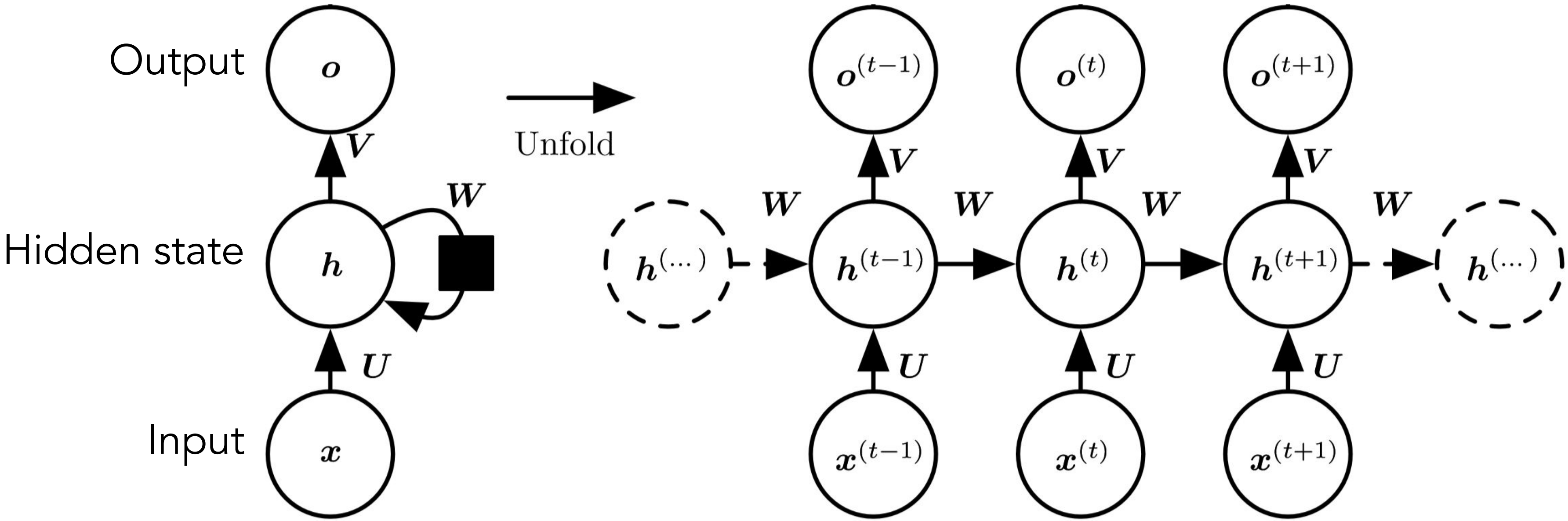


Recurrent Neural Networks (RNNs)

RNN Cell



RNN Graph



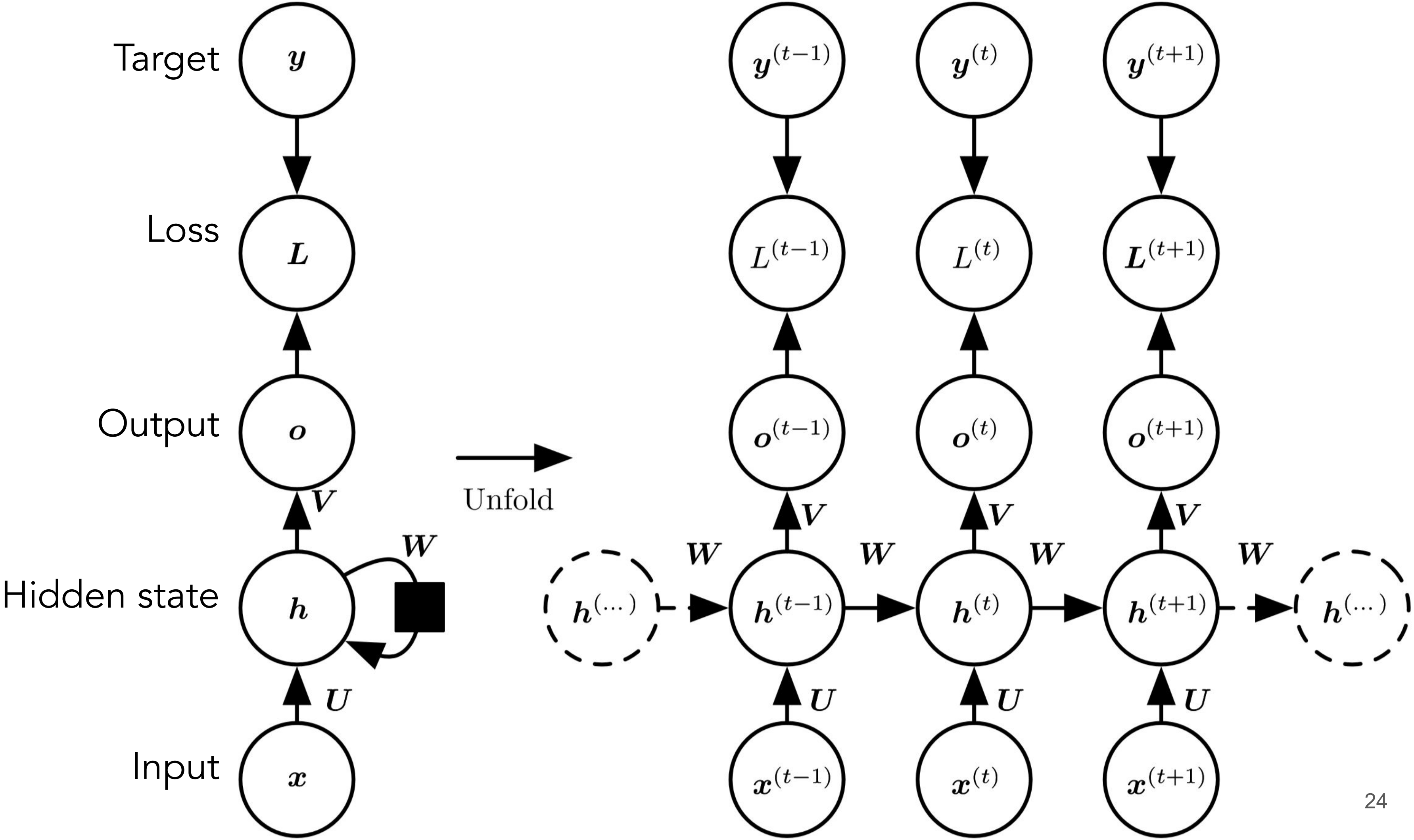
Output
Hidden state
Input
o
h
x
V
W
U
Unfold
$o^{(t-1)}$
$o^{(t)}$
$o^{(t+1)}$
$h^{(\ldots)}$
$h^{(t-1)}$
$h^{(t)}$
$h^{(t+1)}$
$h^{(\ldots)}$
$x^{(t-1)}$
$x^{(t)}$
$x^{(t+1)}$

RNN Feedforward

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

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

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


Target
Loss
Output
Hidden state
Input
y
L
o
h
x
V
W
U
Unfold
$y^{(t-1)}$
$y^{(t)}$
$y^{(t+1)}$
$L^{(t-1)}$
$L^{(t)}$
$L^{(t+1)}$
$o^{(t-1)}$
$o^{(t)}$
$o^{(t+1)}$
$h^{(\dots)}$
$h^{(t-1)}$
$h^{(t)}$
$h^{(t+1)}$
$h^{(\dots)}$
$x^{(t-1)}$
$x^{(t)}$
$x^{(t+1)}$

PyTorch Tutorial - CNN

https://tinyurl.com/2j27t5se

Eboard positions available!

https://forms.gle/aV12v3iJVMnRb1xo6