L9: Deep Neural Network

Shan Wang

Lingnan College, Sun Yat-sen University

2020 Data Mining and Machine Learning LN3119

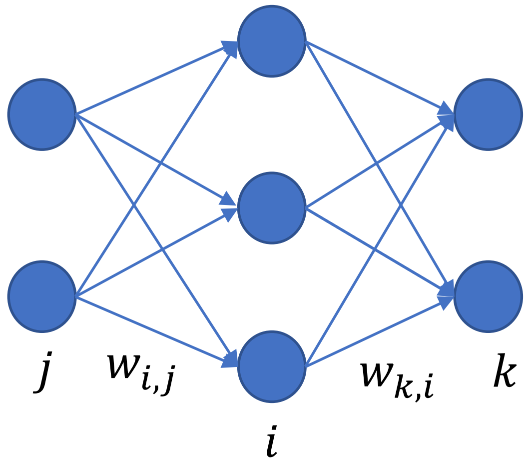
https://wangshan731.github.io/DM-ML/

Last lecture

- One-layer Perceptron
  - $o(\boldsymbol{x}) = f(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$
- Multi-layer Perceptron
  - Model: input, hidden, output
  - Strategy: minimize error
  - Algorithm: BP algorithm
    - ${w_{i,j}}' = w_{i,j} - \eta \varepsilon_i h_j$
    - $\varepsilon_i = \sum_{k=1}^{K} \varepsilon_k \, w_{k,i} {f'}^{i}$
- Overfitting
- Application

Forward Propagation of Info.

Backward Propagation of Error



Course Outline

- Supervised learning
  - Linear regression
  - Logistic regression
  - SVM and kernel
  - Tree models
- Unsupervised learning
  - Clustering
  - PCA
  - EM
- Deep learning
  - Neural networks
  - Convolutional NN
  - Recurrent NN
- Reinforcement learning
  - MDP
  - ADP
  - Deep Q-Network

This lecture

- Deep Learning
- Deep Auto Encoder
- Convolutional NN
- Recurrent NN

Reference: VE 445, Shuai LI (SJTU)

Deep Learning

Deep Learning

- "Deep": the structure is deep, contains many layers

- "Learning"
  - Supervised learning: input data has label
    - Multi-Layer Perceptron, CNN, RNN
  - Unsupervised learning: input data has no label
    - Deep Auto-Encoder
  - Reinforce learning: use penalty and reward
    - FNN and RNN applied in RL, Deep Q-Network

Why we need “deep”?

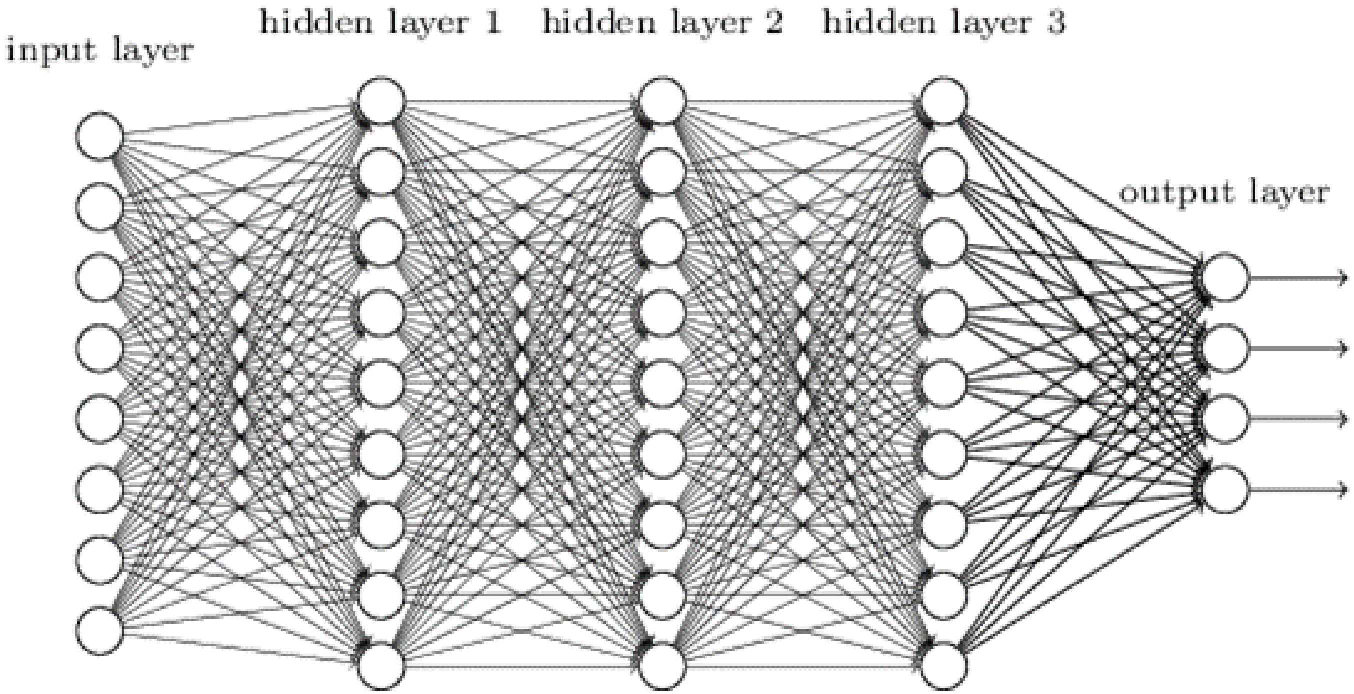
- With more hidden layers, NNs are expected to describe the reality better
  - Functions that can be compactly represented by a depth K architecture, might require an exponential number of neurons to be represented by a depth K-1 architecture
  - Successive layers can learn higher-level features

Problems from Depth

- Lack of big data
  - Now we have a lot of big data
- Lack of computational resources
  - Now we have GPUs and HPCs
- Local optimality
  - Add momentum, pre-training techniques & various optimization algorithms
- Gradient vanishing
  - Use ReLU activation function…
- Too many parameters
  - Train the network layer by layer & dropout

Multi-Layer Perceptron

- MLP is a fully connected neural network with multiple hidden layers

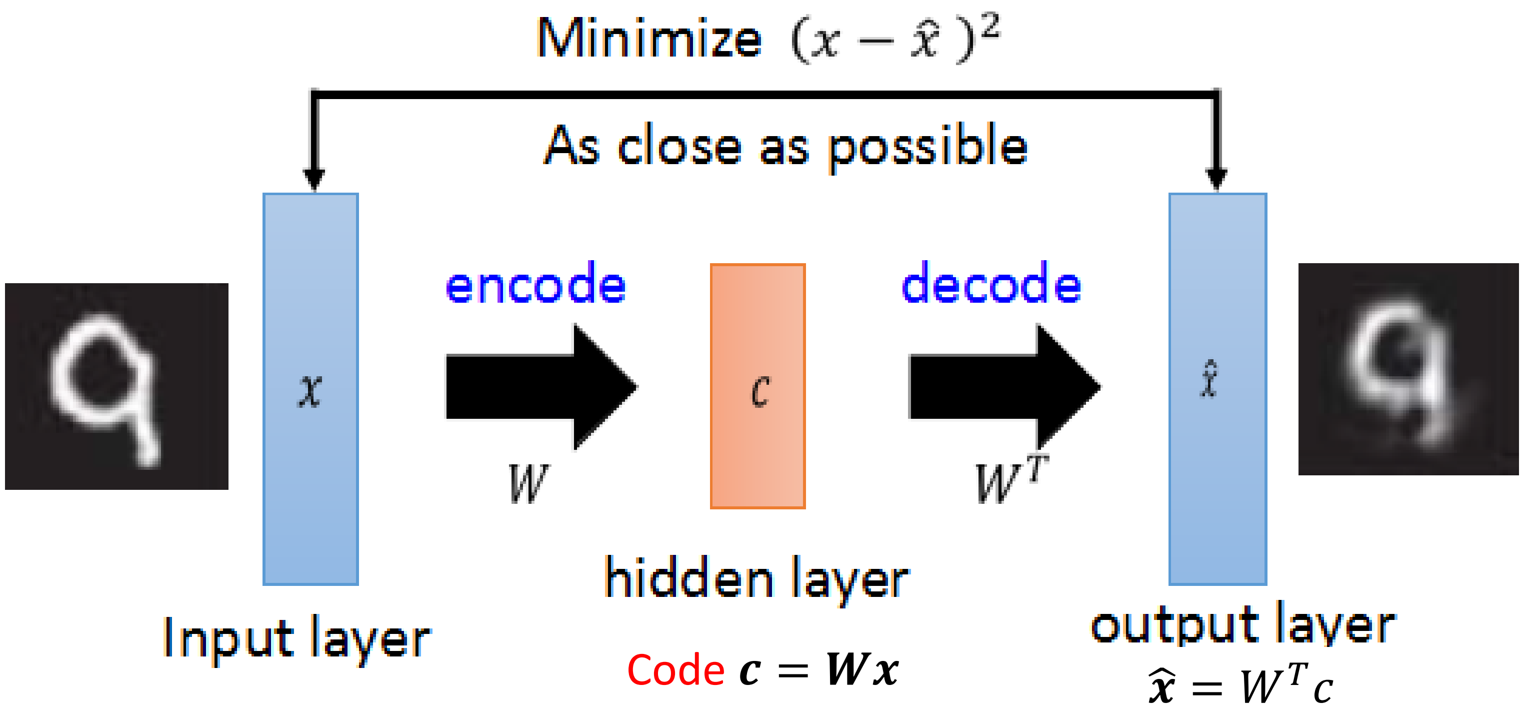


Deep Auto-Encoder

Auto-Encoder

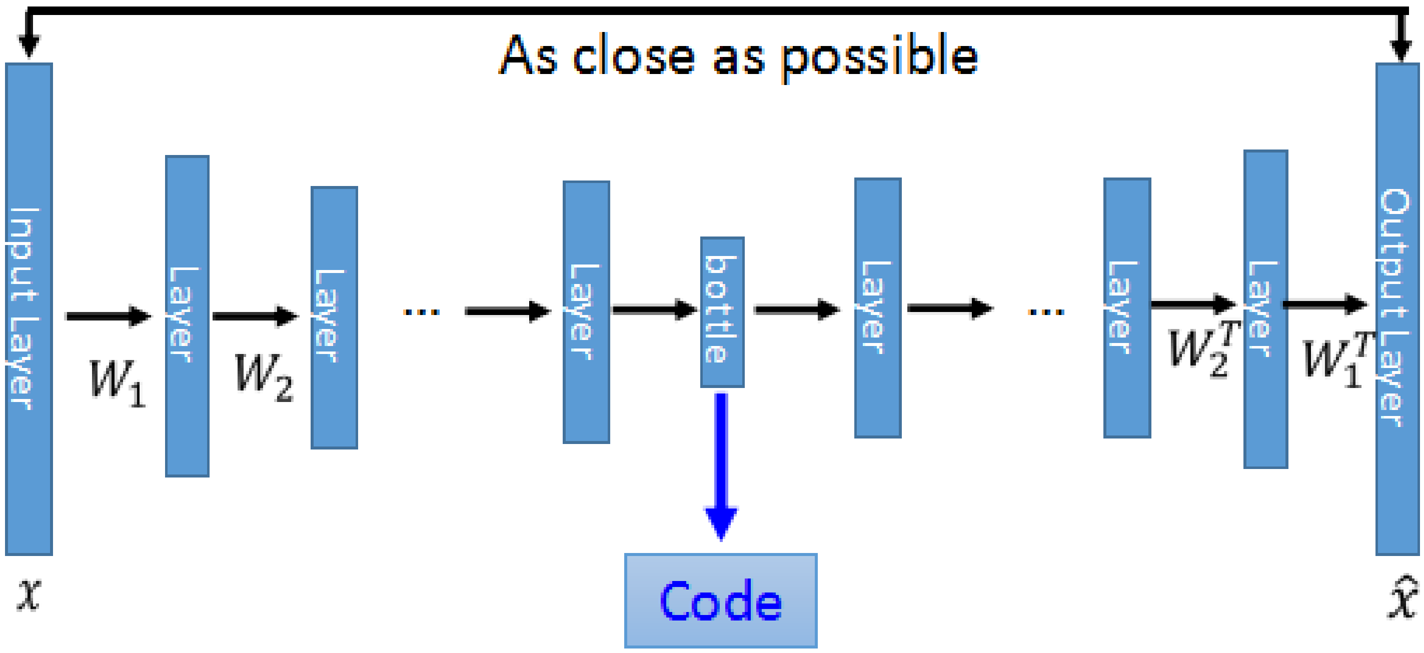
- Auto-Encoder can be used for data suppress, dimensionality reduction, pre-training NNs etc.
- An auto-encoder contains an encoder and a decoder
- The encoder plays the role of coding the original data
  - Data A $\longrightarrow$ Codes
- The decoder plays the role of decoding the “code”
  - Codes $\longrightarrow$ Data B
- Good Auto-Encoder:
  - Data B is very close to Data A

Linear Auto-encoder



Deep Auto-Encoder

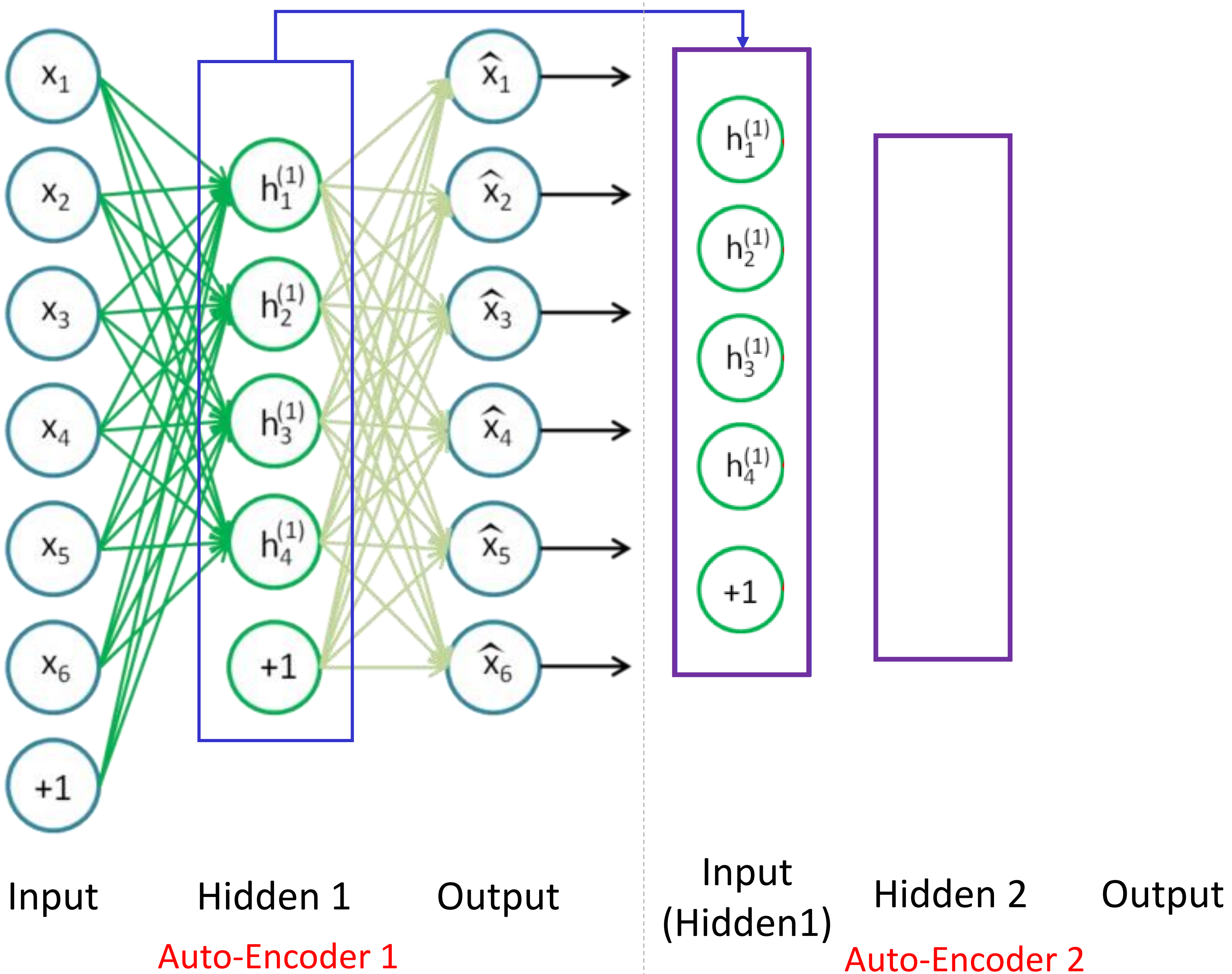
- Of course, the Auto-Encoder can be very deep
  - Not necessary to be symmetric



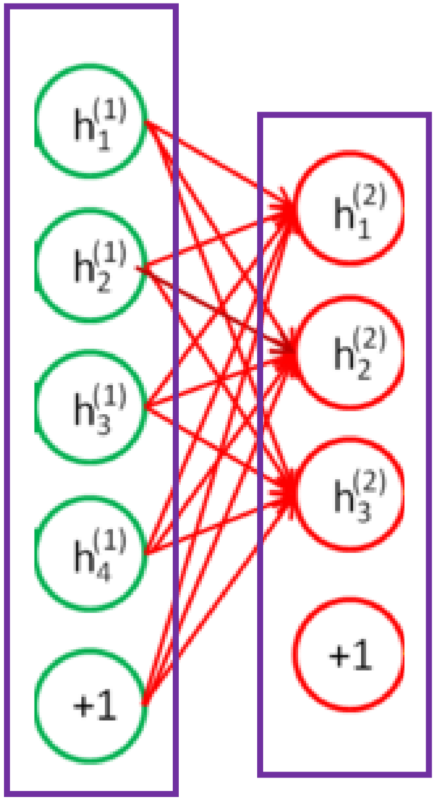
Stacked Auto-Encoder

- Stacked Auto-Encoder can be used to pre-train the deep network layer by layer
- Two steps
  - Unsupervised pre-training for feature layers
    - Start with the 1st hidden layer, use Auto-Encoder to train this layer to make its inputs and outputs consistent
    - and then use the “code” in the 1st hidden layer as the inputs of the 2nd hidden layer, and repeat…
  - Supervised fine-tuning for classification
    - Compose these pre-trained hidden layers
    - add an output layer at the last, and train the output layer or fine-tune the whole network

Deep Auto-Encoder



Trained Deep Neural Network



Hidden 1 Hidden 2

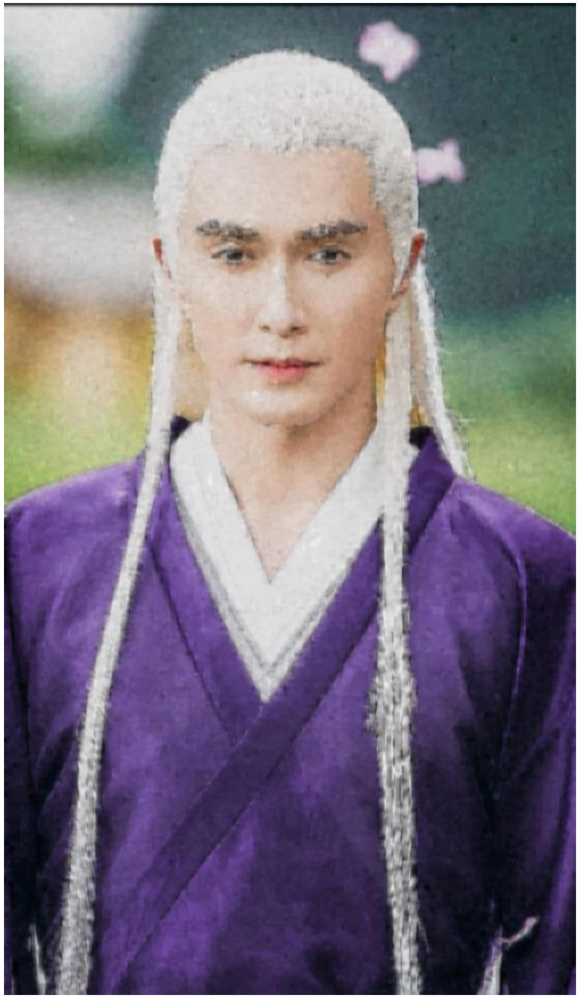
Convolutional NN

Convolutional Neural Network

- CNN is widely used for image recognition
- It is an end-to-end recognition system
  - A non-linear map that takes raw pixels directly to labels
- Contains the following layers with flexible order and repetitions
  - Convolution layer
  - Activation layer (ReLU: $\max\{0, x\}$)
  - Pooling layer

What is Convolution?

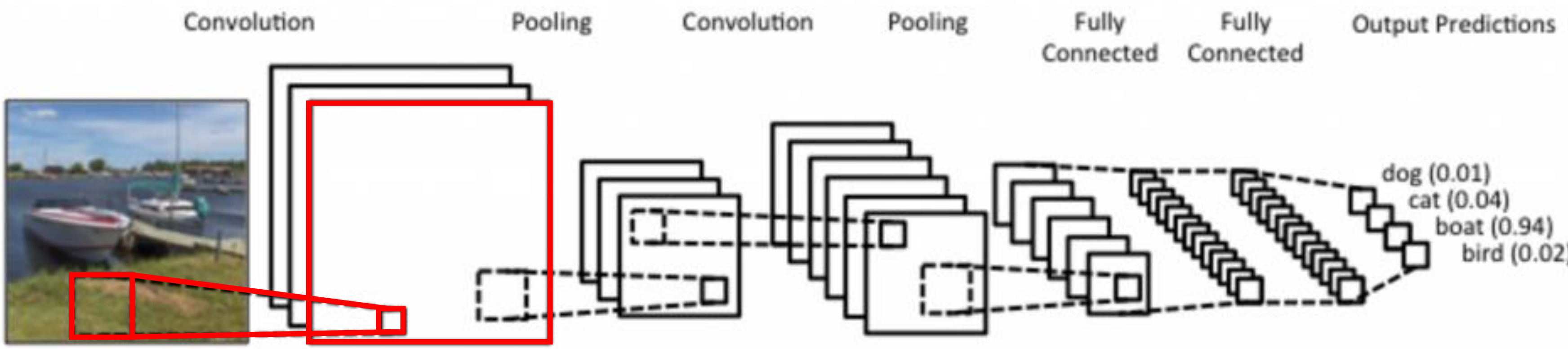
东华帝君

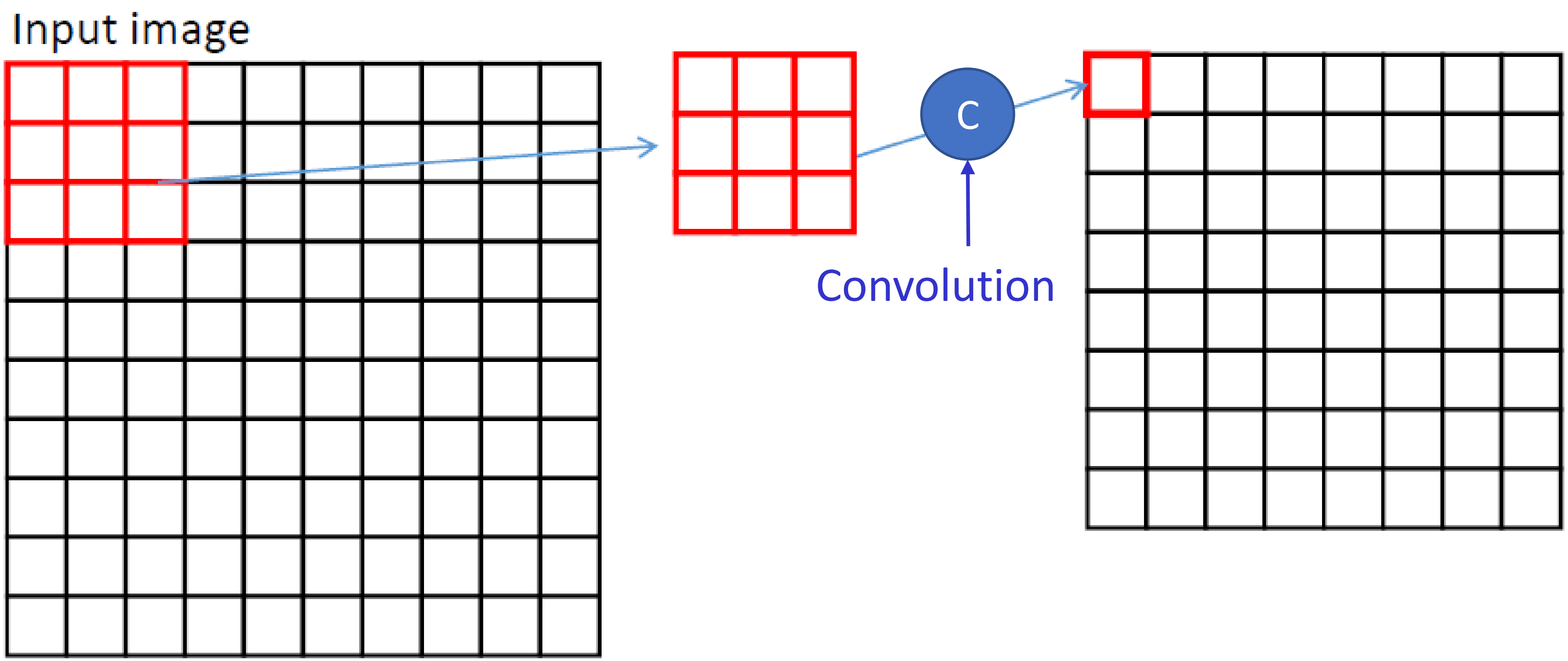


模糊和锐化后的东华帝君

Picture credit: @是麻辣兔兔

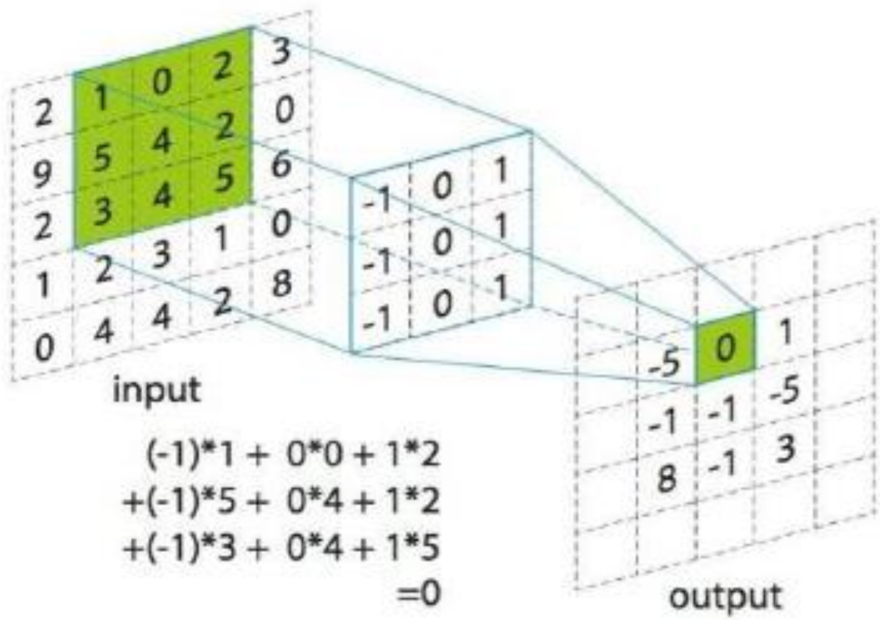
Structure


Convolution
Pooling
Convolution
Pooling
Fully Connected
Fully Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)


Input image
C
Convolution

Convolution in Neural Networks

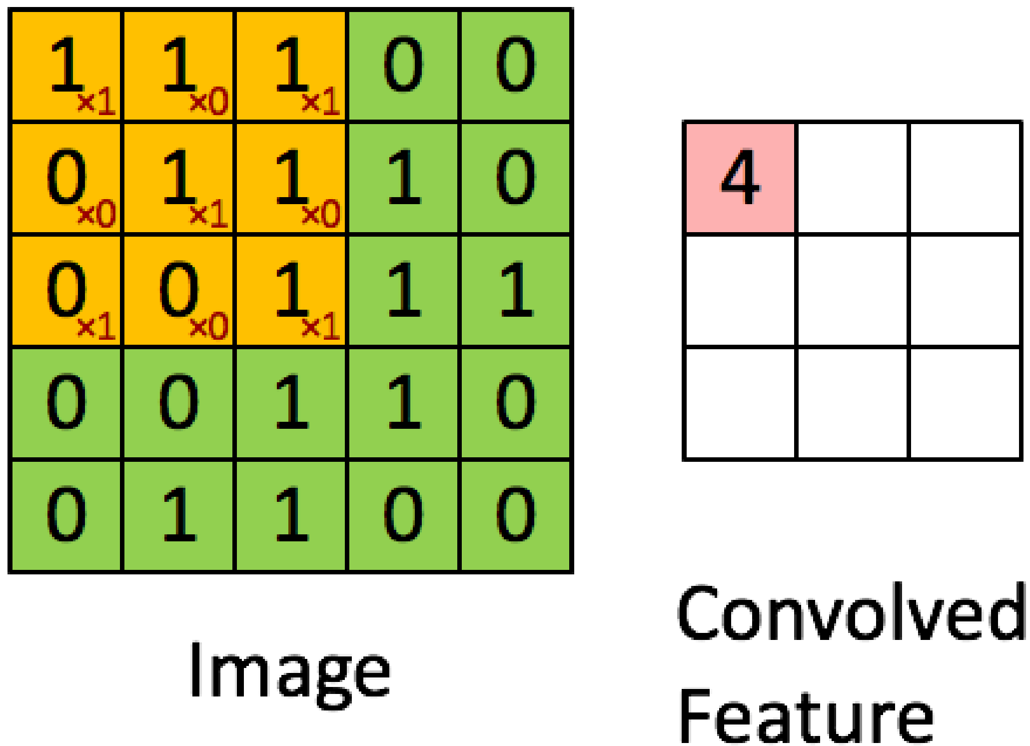
- Given an input matrix (e.g. an image)
- Use a small matrix (called filter or kernel) to screening the input at every position of the input matrix
- Put the convolution results at corresponding positions



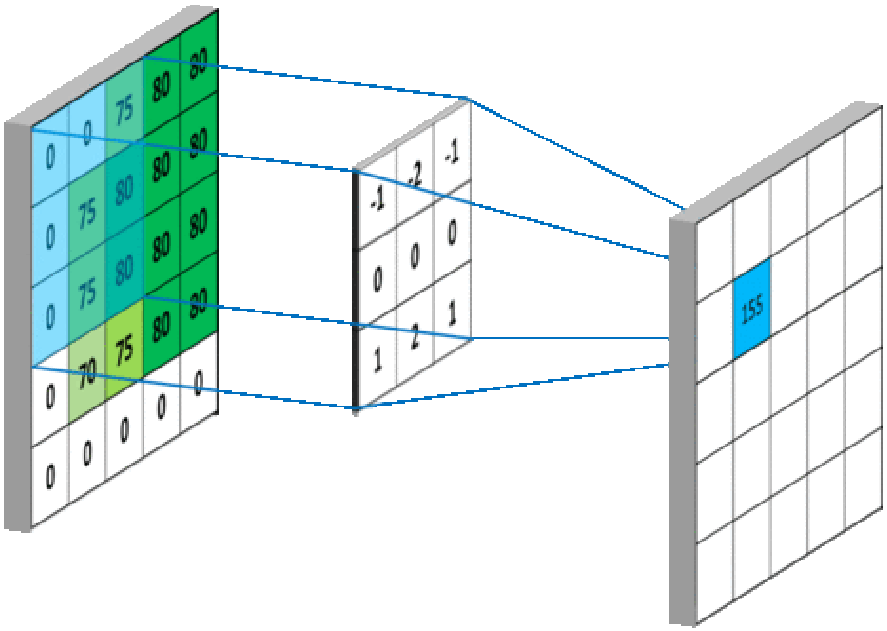
Sometimes, we add ReLU activation layer after the outputs

$$output' = \max\{0, output\}$$

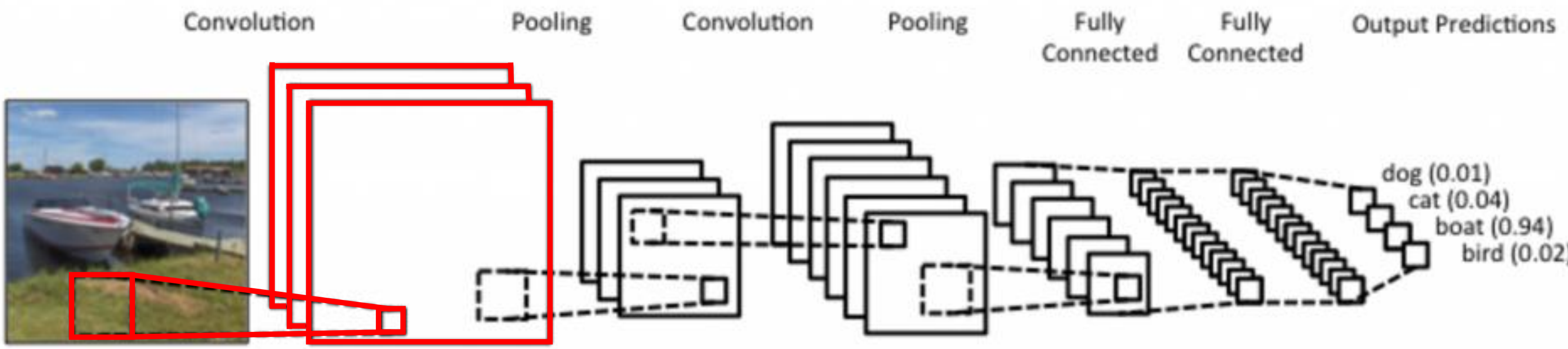
Convolutions Visualization

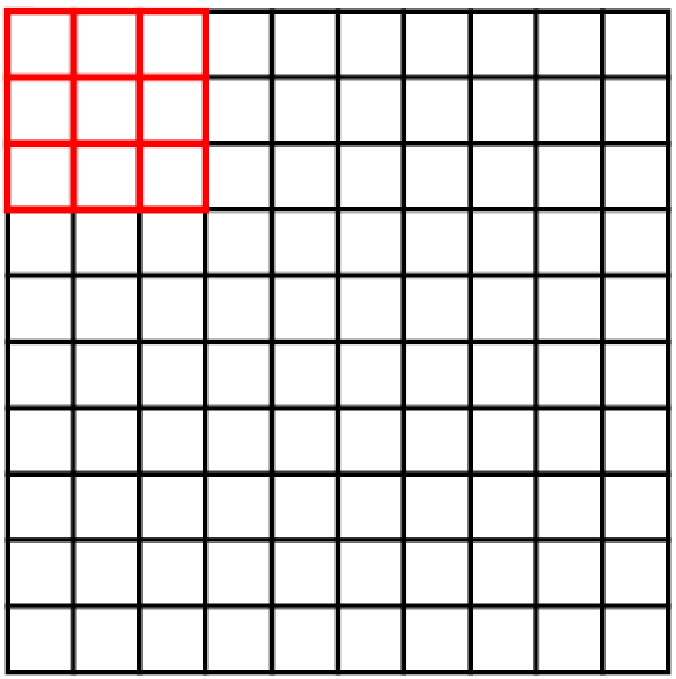


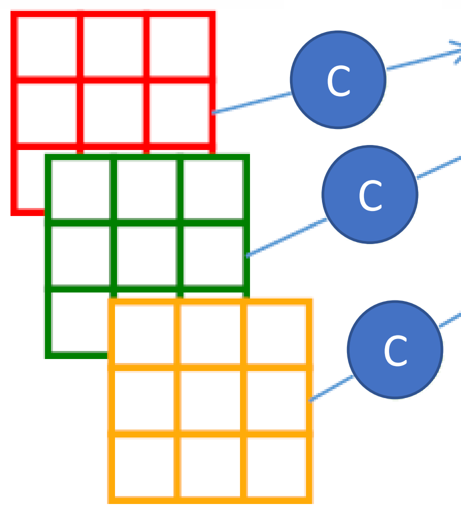
Convolutions Visualization (cont.)

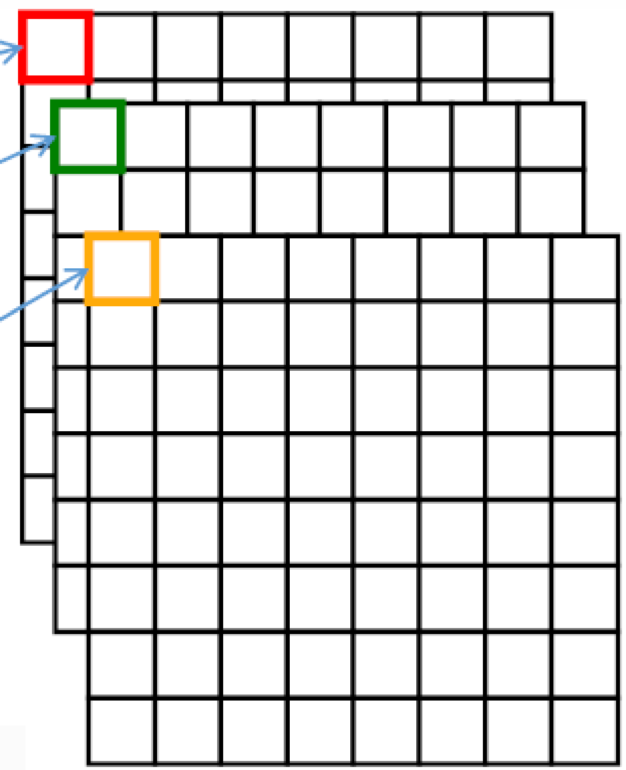


Kernel Numbers





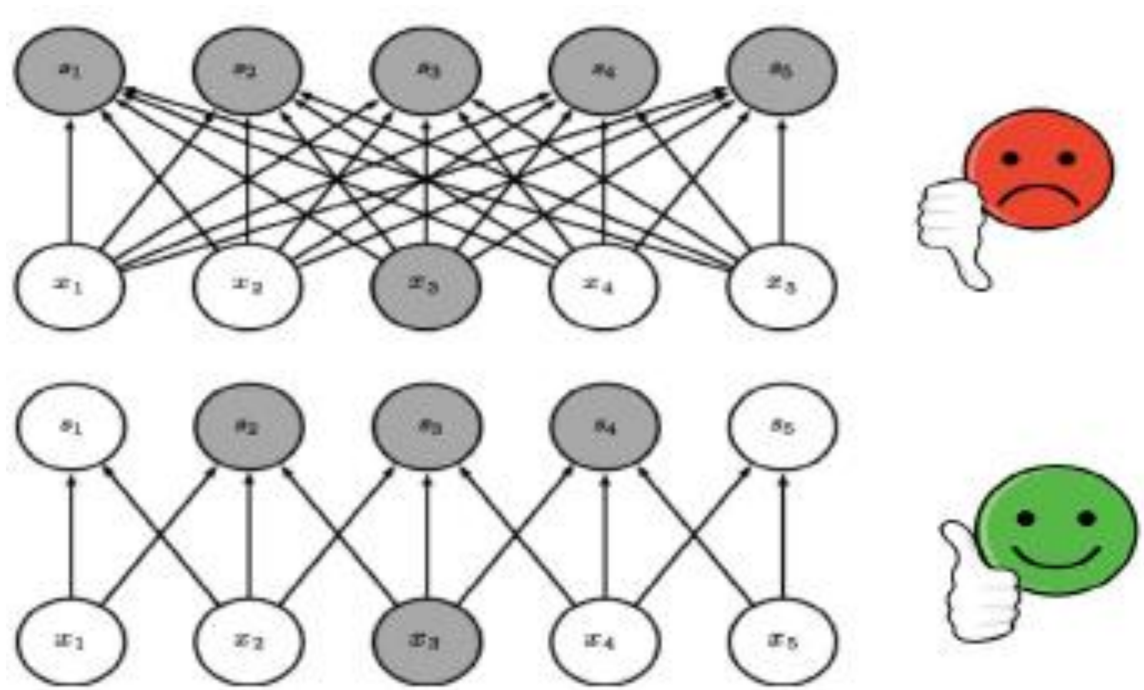




Decide the number of outputs

Why Convolutions?

- A local in a picture is more important
  - Sparse connections
  - Less computing burden

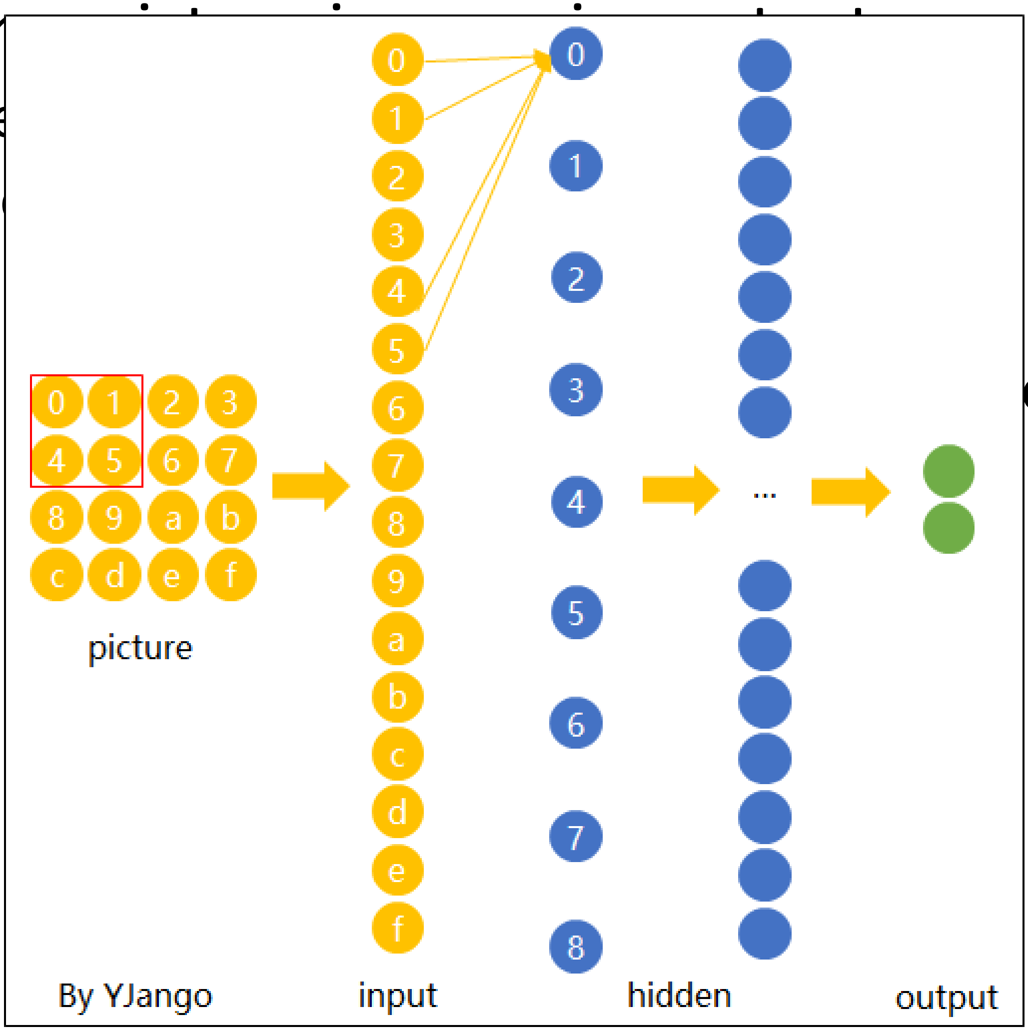


Why Convolutions?

- A local in a picture is more important
  - Sparse connections
  - Less computing burden
- Position invariance
  - A dog is a dog no matter where he is in the picture
  - Share convolution kernel
  - Share weights

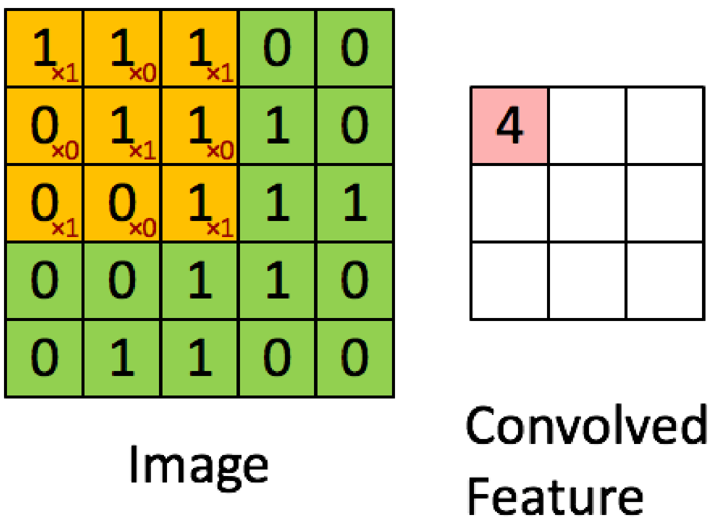
Why Convolutions?

- A local i
  - Sparse
  - Less c
- Position
  - A dog ... cture
  - Share
  - Share



Important Parameters

- Kernel size
  - The dimension of kernel matrix
- Stride
  - The distance that the filter is moved in each step



Kernel Size:

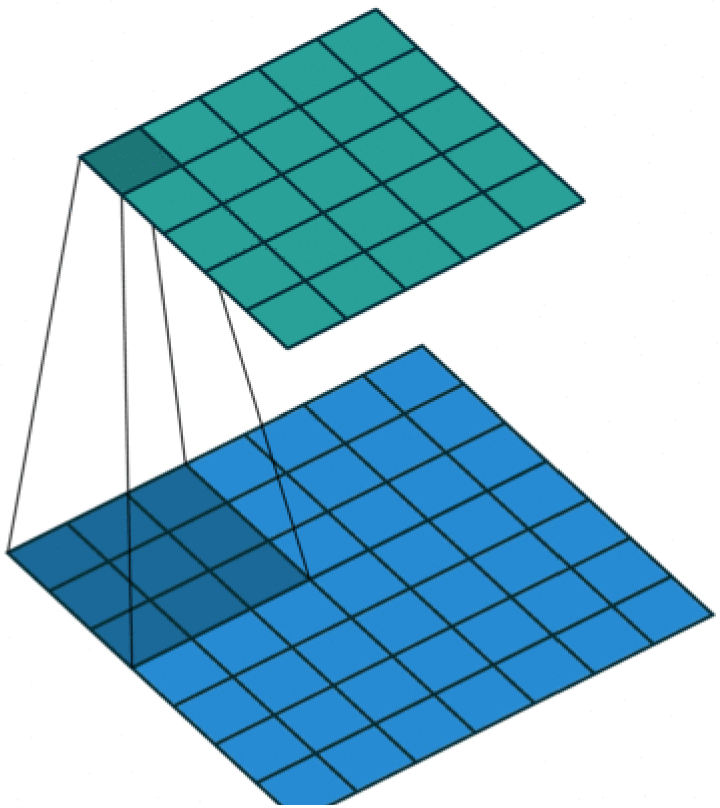
Stride:

Important Parameters

- Kernel size
  - The dimension of kernel matrix
- Stride
  - The distance that the filter is moved in each step
- Pad
  - Add numbers (usually 0) around the input data

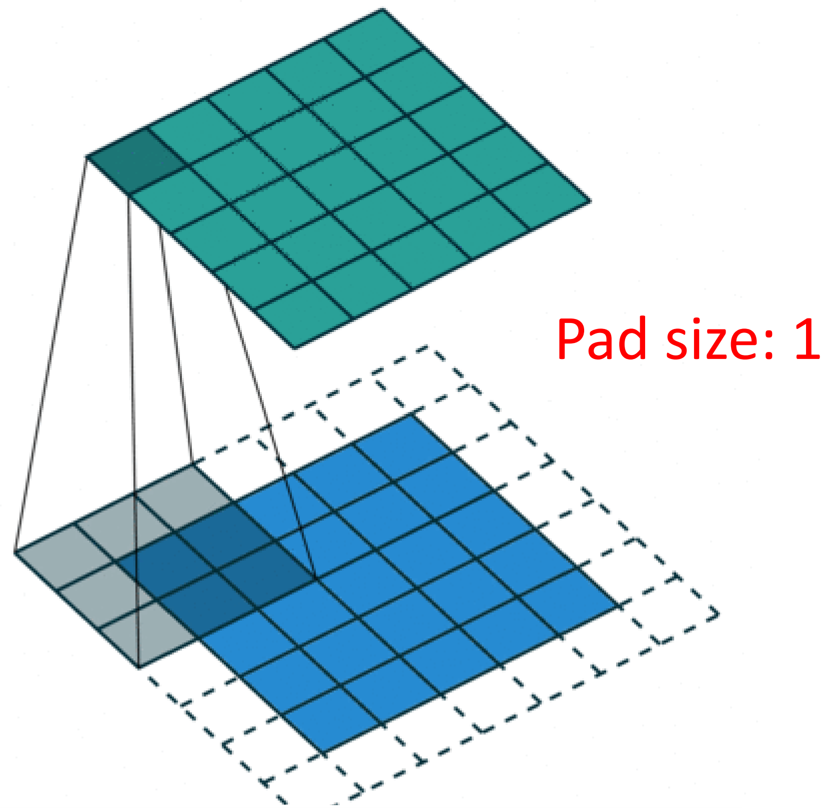
Padding

Without padding



7 X 7 ⇒ 5 X 5

With padding



5 X 5 ⇒ 5 X 5

Parameters about Convolution Layer

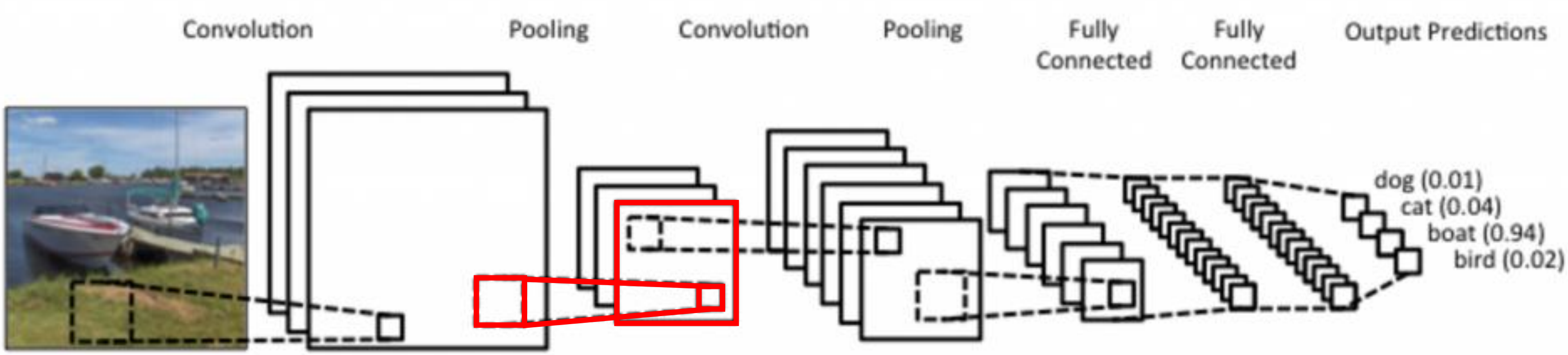
- Kernel size: k
  - The dimension of kernel matrix is $k \times k$
- Stride: s
  - The distance s that the filter is moved in each step
- Pad: p
  - Add p round of numbers (usually 0) around input data

- Output size calculator:
  - Input size: $w \times h$
  - Output size:

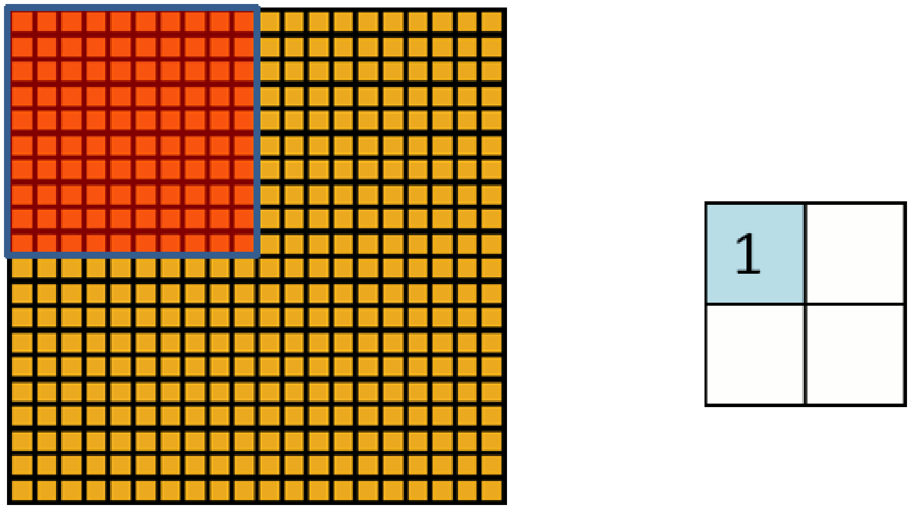
$$w' = {}^{w+2p-k}/_{s} \qquad h' = {}^{h+2p-k}/_{s}$$

Pooling Layer

- Make the representations denser and more manageable
- Operate over each activation map independently



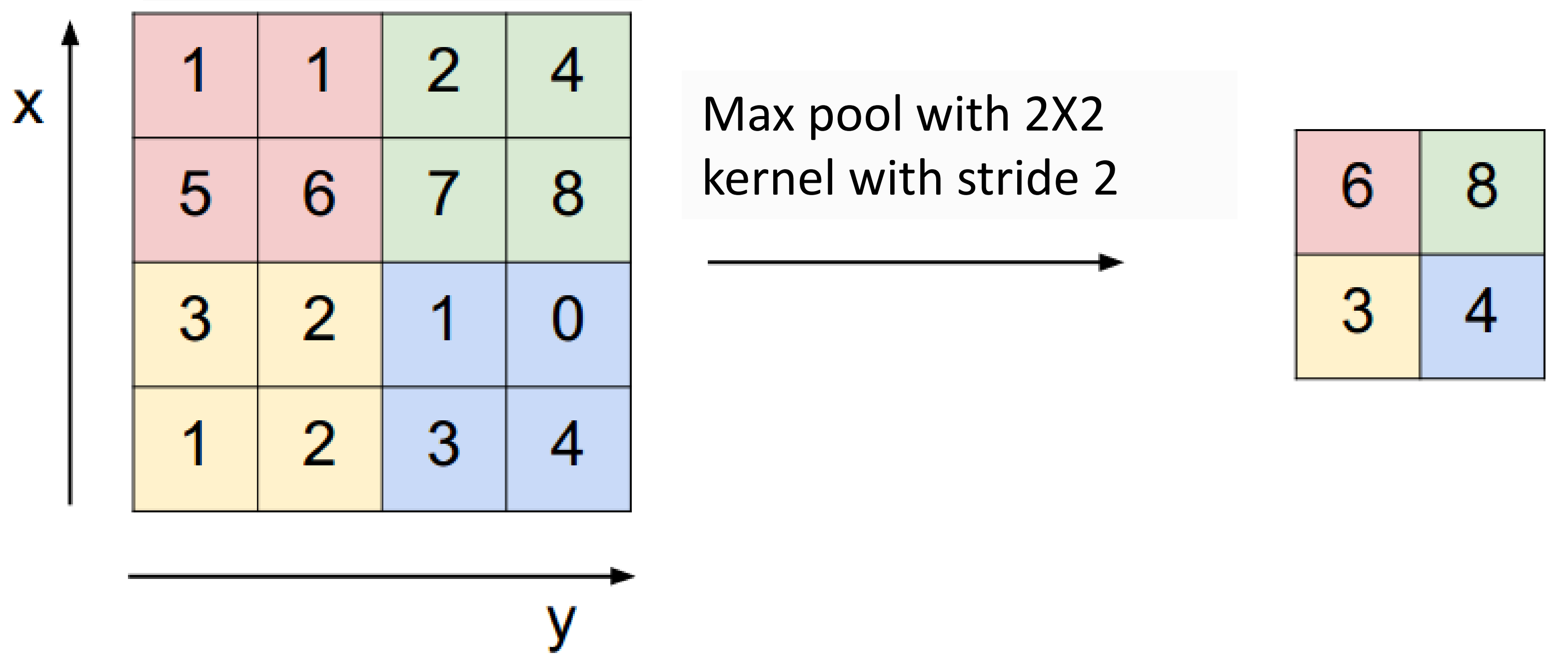
Pooling



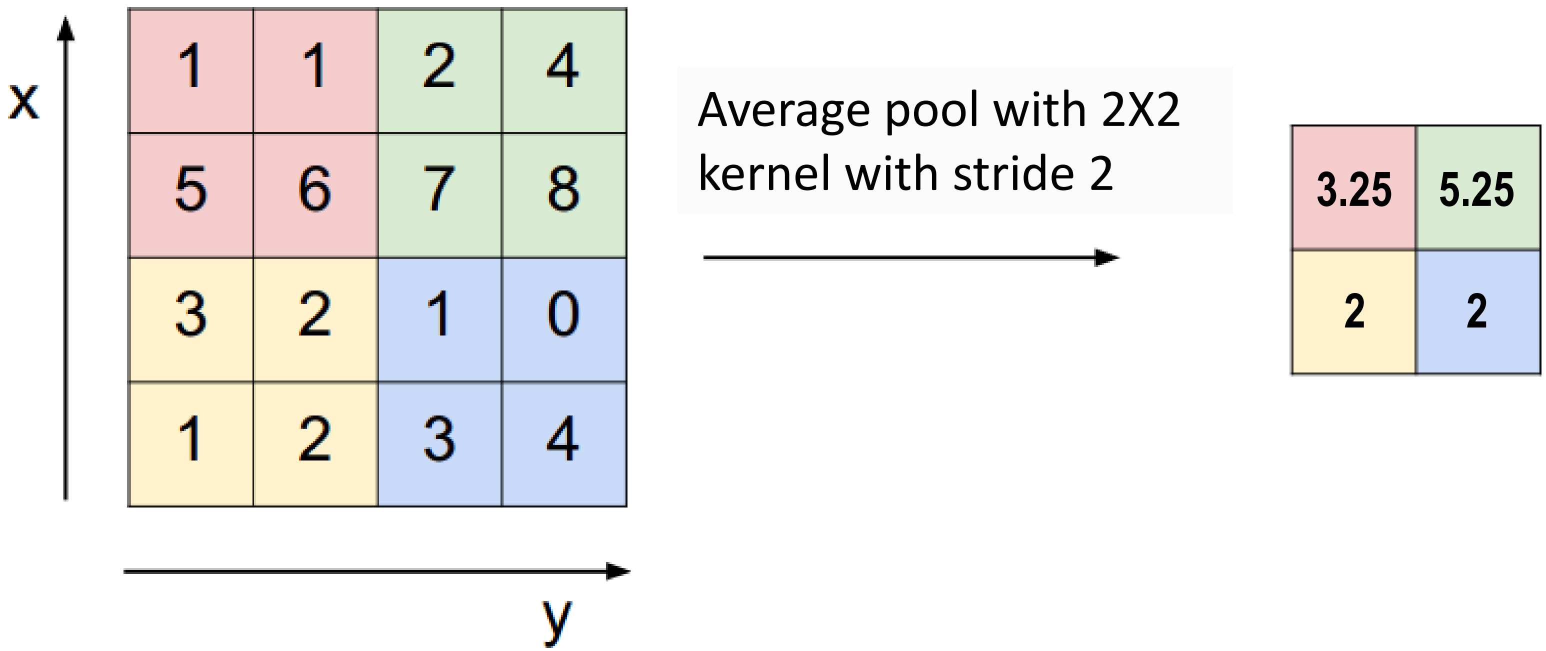
Convolved feature

Pooled feature

Max Pool


x
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
y
Max pool with 2X2
kernel with stride 2
6 8
3 4

Average Pool



Recurrent NN

Motivation of RNN

- In traditional NN
  - Assume all inputs and outputs are independent of each other
  - Input and output length are fixed
- But this might be bad for some tasks
  - Predict next word in a sentence
    - "Context": You better know which words came before it
- Recurrent
  - Perform the same task for every element of a sequence, with the output being dependent on the previous computations
- They have a "memory" which captures information about what has been calculated so far

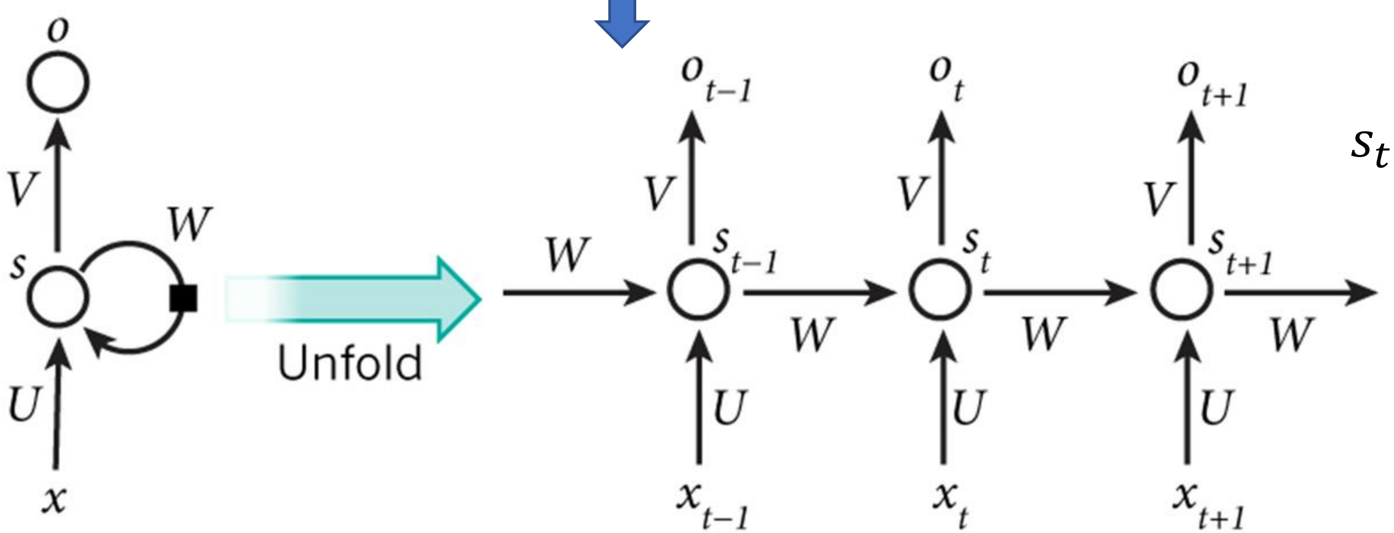
RNN

Two-layer feedforward network

x: input vector
o: output vector
s: hidden state vector
U: layer 1 param. matrix
V: layer 2 param. Matrix
f: tanh or ReLU

$$s = f(Ux), o = f(Vs)$$

Add time-dependency of the hidden state s



$$s_t = f(Ux_t + Ws_{t-1})$$
$$o_t = f(Vs_t)$$

W: State transition param. matrix

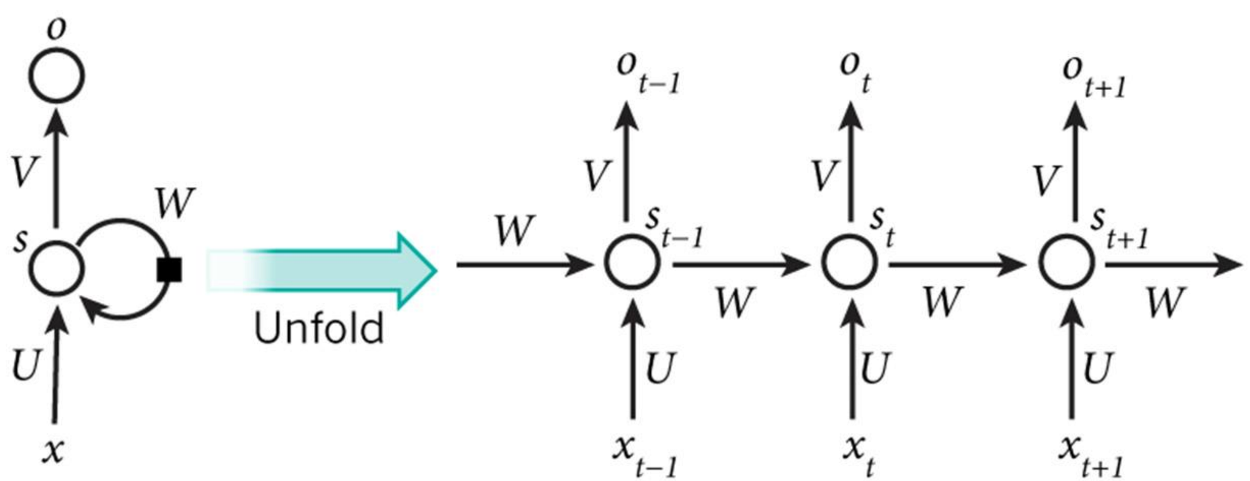
RNN

- x_t is the input at time t
- s_t is the hidden state at time t
  - It is the "memory" of the network
  - Is calculated based on previous hidden state and the input at the current step

$$s_t = f(Ux_t + Ws_{t-1})$$

- o_t is the output at time t

E.g. If we want to predict the next word in a sentence, o_t is a vector of probabilities over certain vocabulary



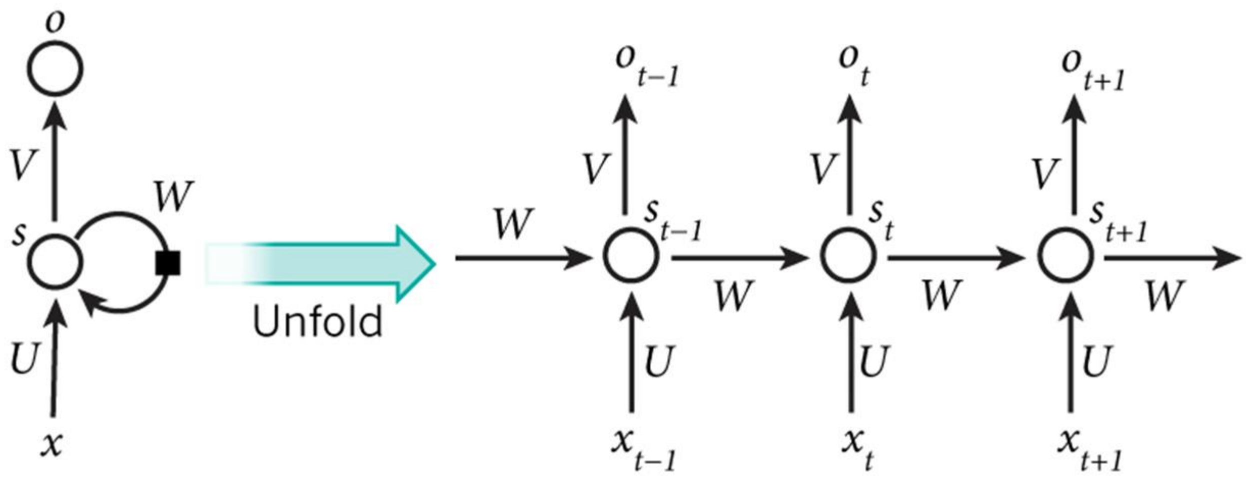
RNN Features

- s_t is the "memory" of the network
- o_t is based on the memory at t
- RNN share weights U and W
  - Reduce computation complexity
- The output at each time step might be unnecessary
  - E.g. When predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word
- The input at each time step might be unnecessary
- Most important feature:

  The hidden state captures some information about a sequence

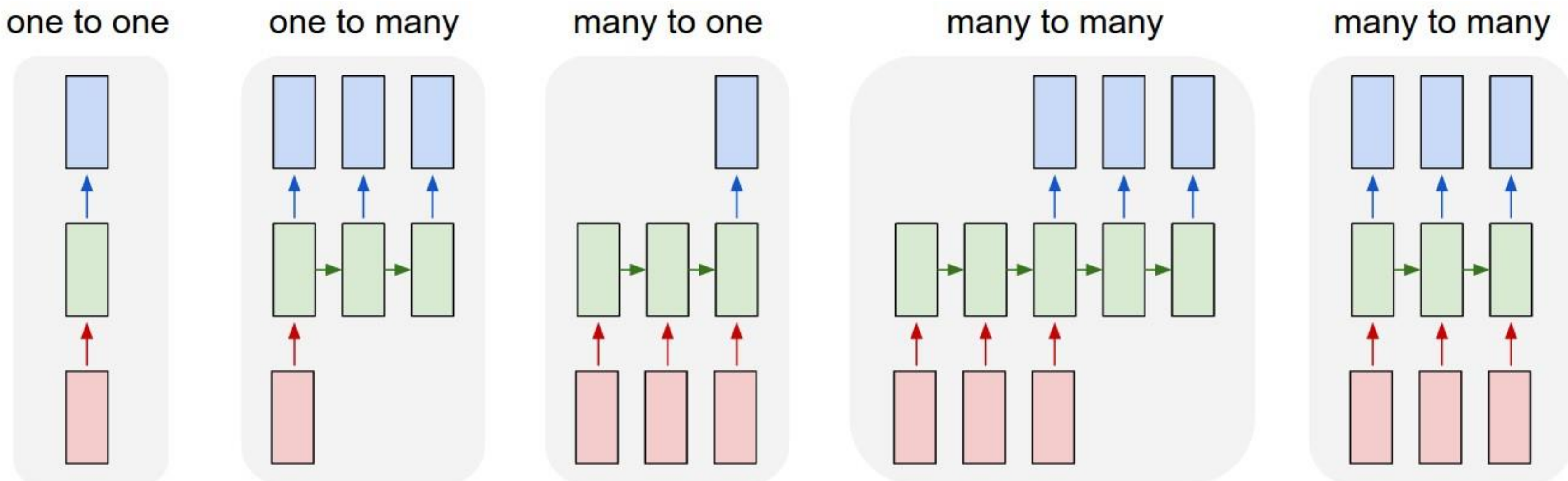
Strategy and Algorithm

- Strategy: minimize the cross entropy
  - E.g., $\hat{y}_t = softmax(o_t)$, the prediction
  - y_t is the correct word at time t
  - Loss: $-\sum_t \ y_t \log \hat{y}_t$
- Algorithm: Backpropagation Through Time (BPTT)
  - E.g., in order to calculate the gradient at $t = \ 4$, we would need to backpropagate 3 steps and sum up the gradients



Different RNN

- Different architecture for various tasks



Traditional NN | Image captioning, Text generation | Text classification, Sentiment analysis | Machine translation, Dialogue system | Stock price estimation, Video frame classification

- Strongly recommend Andrej Karpathy's blog
  - http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Summary

- Universal Approximation: two-layer neural networks can approximate any functions
- Backpropagation is the most important training scheme for multi-layer neural networks so far
- Deep learning, i.e. deep architecture of NN trained with big data, works incredibly well
- Neural networks built with other machine learning models achieve further success

Lecture 9 Wrap-up

✓Deep Learning

✓Deep Auto Encoder

✓Convolutional NN

✓Recurrent NN

Next Lecture

- Supervised learning
  - Linear regression
  - Logistic regression
  - SVM and kernel
  - Tree models
- Unsupervised learning
  - Clustering
  - PCA
  - EM
- Deep learning
  - Neural networks
  - Convolutional NN
  - Recurrent NN
- Reinforcement learning
  - MDP
  - ADP
  - Deep Q-Network

Questions?

Shan Wang (王杉)
https://wangshan731.github.io/