The Hitchhiker's Guide to Deep Learning

v0.1

Cheng PENG

2018.03.13

Content

- Introduction
- Special Networks
  - CNN (Convolutional Neural Network)
  - RNN (Recurrent Neural Network)
- Application
  - SST Prediction

Main Refs:
[1] Deep learning, Yann LeCun, Yoshua Bengio& Geoffrey Hinton, 2015 Nature
[2] Deep Learning Tutorial, 李宏毅
[3] A Tutorial on Deep Learning, Quoc V. Le
[4] 机器学习, 周志华
[5] Deep Learning Book Goodfellow et al. 2016
[6] Prediction of Sea Surface Temperature Using Long Short-Term Memory, IEEE Geoscience and Remote Sensing Letters, 2017

Content

- Introduction
  - Overview
  - Step
  - Application
  - Framework
- Special Networks
  - CNN
  - RNN
- Application
  - SST Prediction

Overview

“Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.”

— 《Deep learning》 Yan LeCun et al. 2015 Nature

Ref:
[1] Deep learning, Yann LeCun, Yoshua Bengio& Geoffrey Hinton, 2015 Nature

Overview

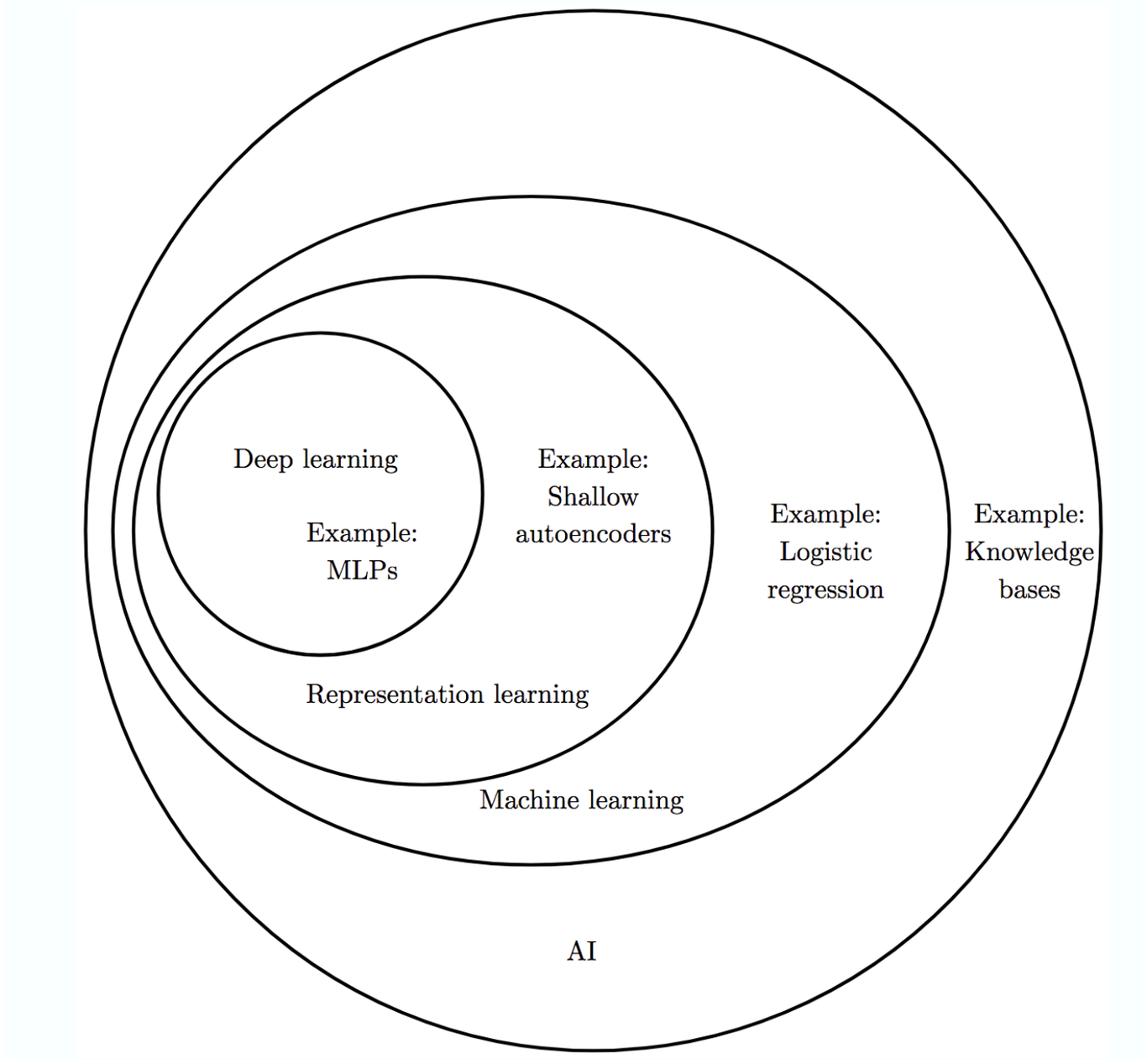
*"Deep learning allows **computational models** that are composed of **multiple processing layers** to learn **representations of data** with multiple levels of abstraction."*

— 《Deep learning》 Yan LeCun et al. 2015 Nature

computational model —> learning (training) result.
multiple processing layers —> feature of the model (deep).
representation of data —> purpose of the model.
data —> input (source).

Ref:
[1] Deep learning, Yann LeCun, Yoshua Bengio& Geoffrey Hinton, 2015 Nature

Overview

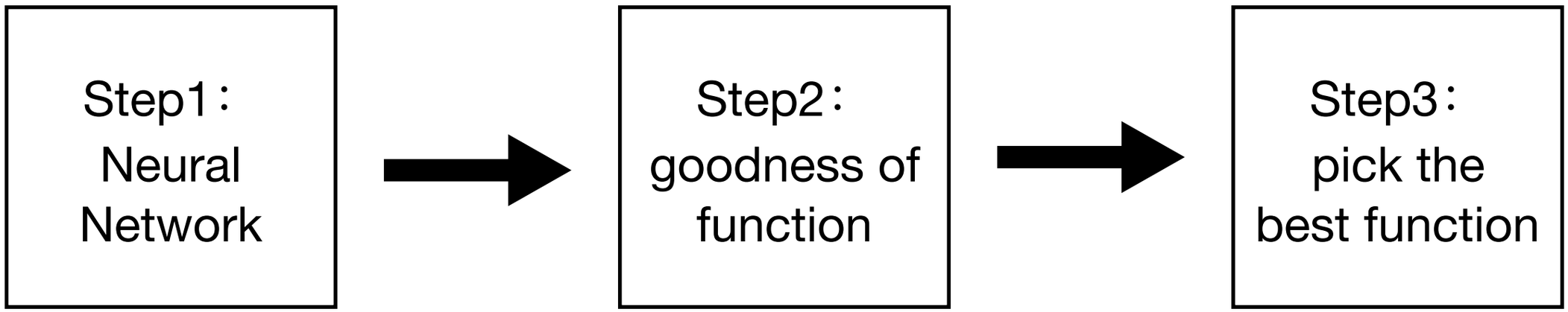


*"A Venn diagram showing how **deep learning** is a kind of **representation learning**, which is in turn a kind of **machine learning**, which is used for many but not all approaches to **AI**. Each section of the Venn diagram includes an example of an AI technology."*

—Deep Learning Book Ian Goodfellow et al.

Ref:
[1] Deep Learning Book Chapter1 Introduction
[2] Feature learning Wikipedia
[3] Deep Learning Book Chapter15 Representation Learning

Step

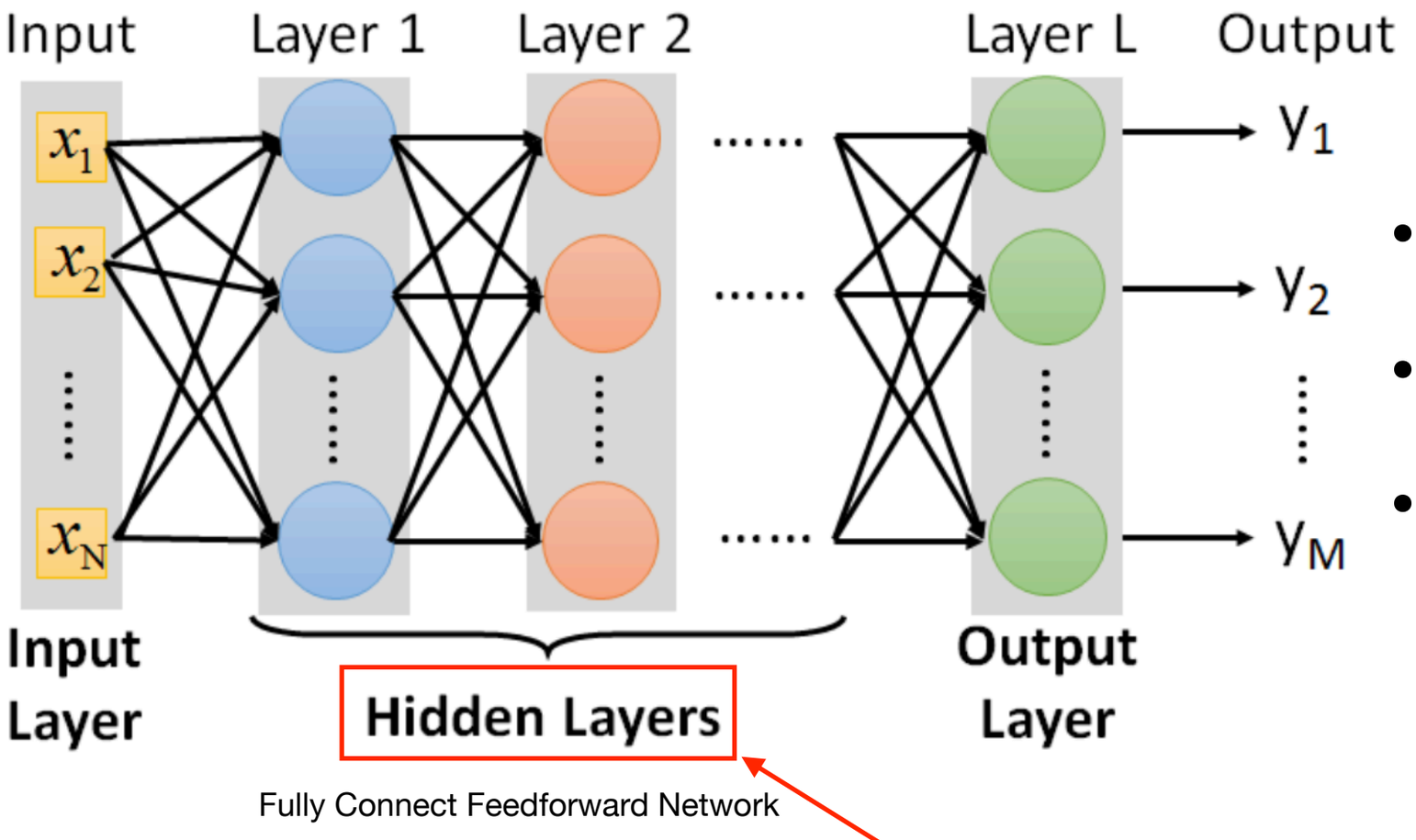


Ref:
[1] Deep Learning Tutorial, 李宏毅

Step

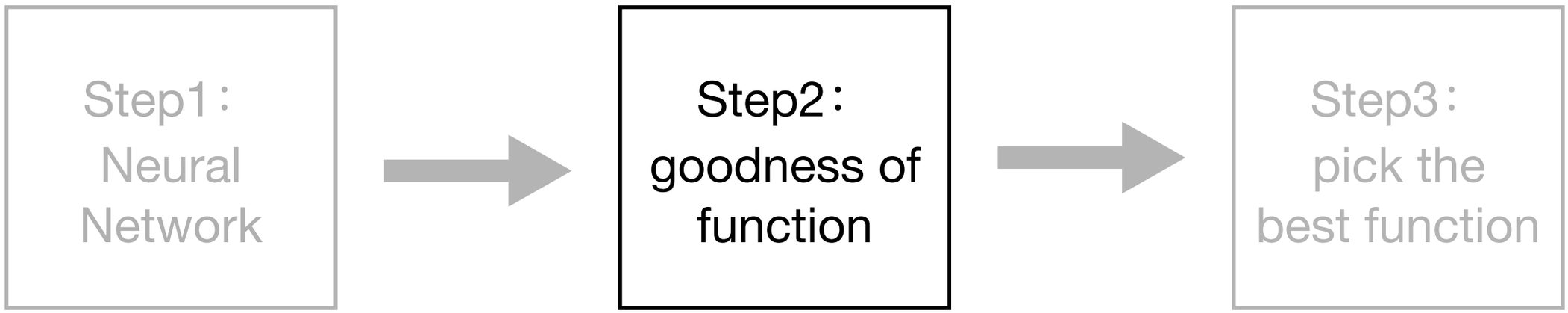

Step1：
Neural Network
Step2：
goodness of function
Step3：
pick the best function

Neural Network

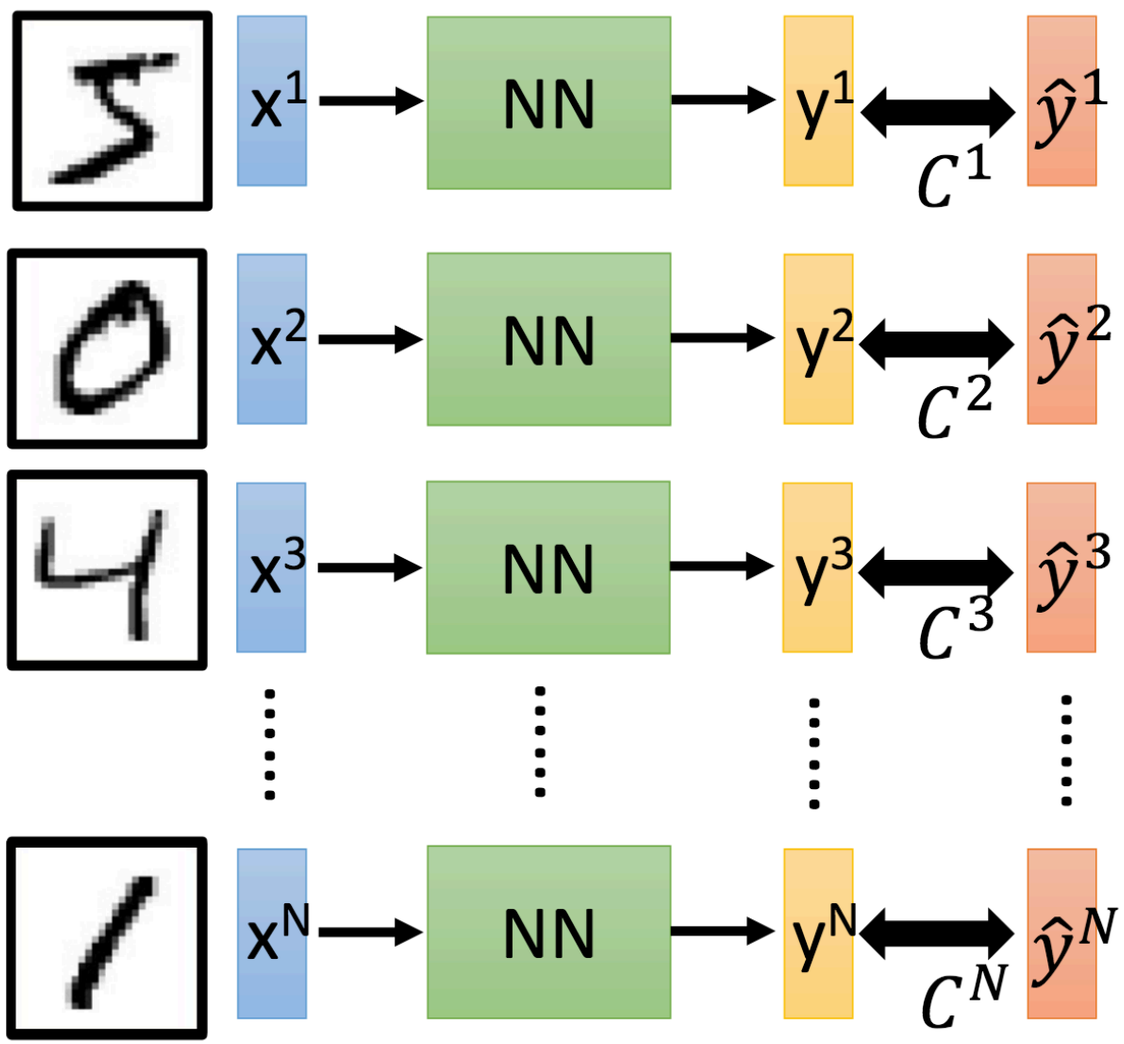


- For a given network structure, it defines a **function set**.
- Each neural have **weights** and **bias** (parameters).
- Input and output are all vectors.

Step


Step1：
Neural
Network
Step2：
goodness of
function
Step3：
pick the
best function

Goodness of function



Total Loss(loss function):

$$L = \sum_{n=1}^{N} C^n$$

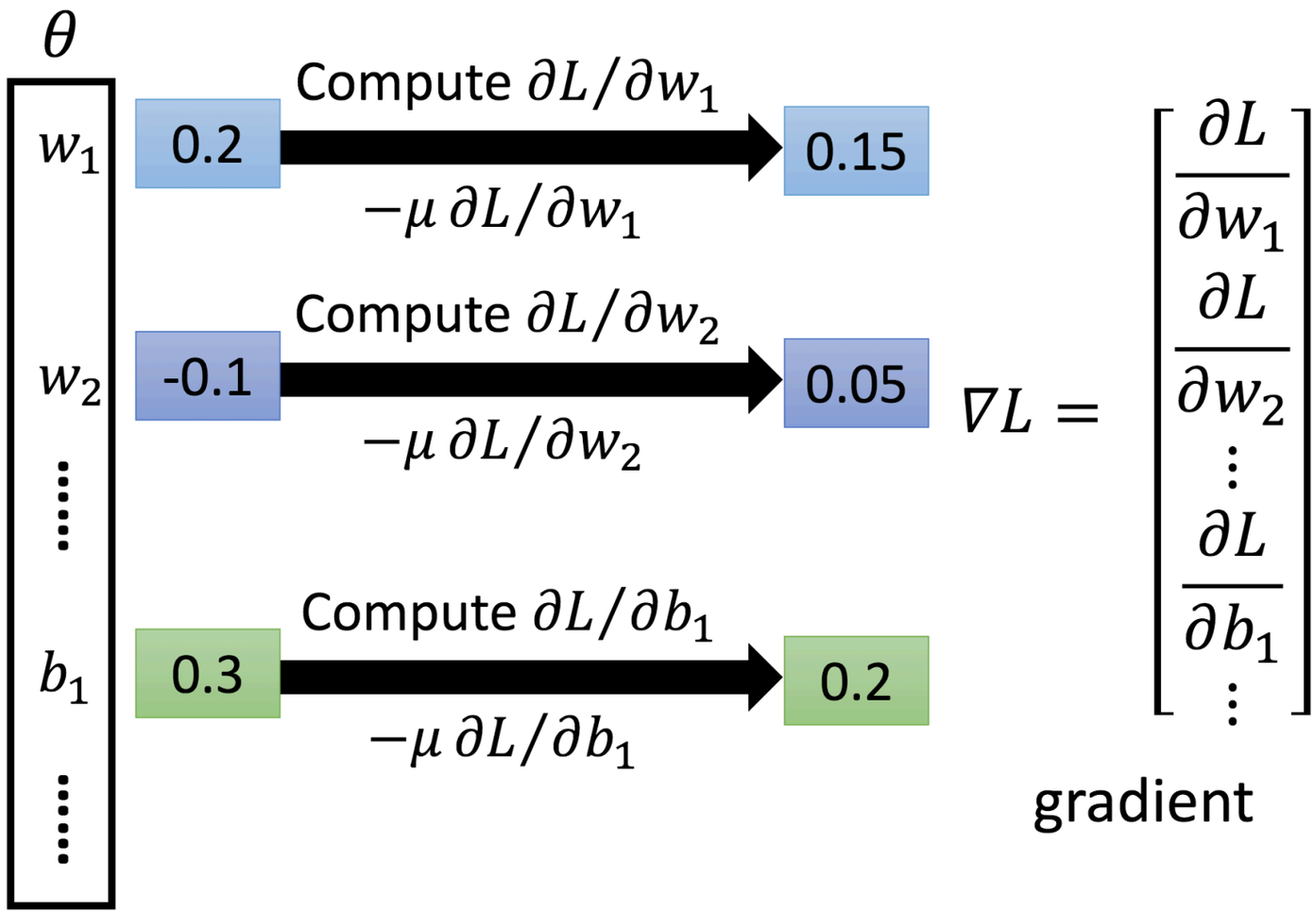
↓

Find **a set of parameters** that minimizes total loss L.

Step


Step1：
Neural Network
Step2：
goodness of function
Step3：
pick the best function

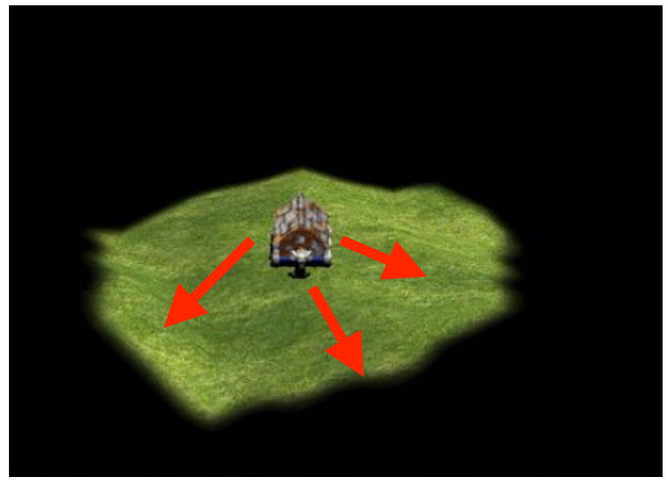
Gradient Descent



$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient

What you are doing:



Hints:

- Tuning your **learning rate**
- Stochastic Gradient Descent
- Mini-batch Gradient Descent
  (get your hands dirty!)

Ref:

[1] 梯度下降法的三种形式*BGD*、*SGD*以及*MBGD*

Application Fields

*“Many experiments have shown that neural networks are particularly good with **natural data** (speech, vision, language) which exhibit **highly nonlinear properties**. ”*

— A Tutorial on Deep Learning, Quoc V. Le

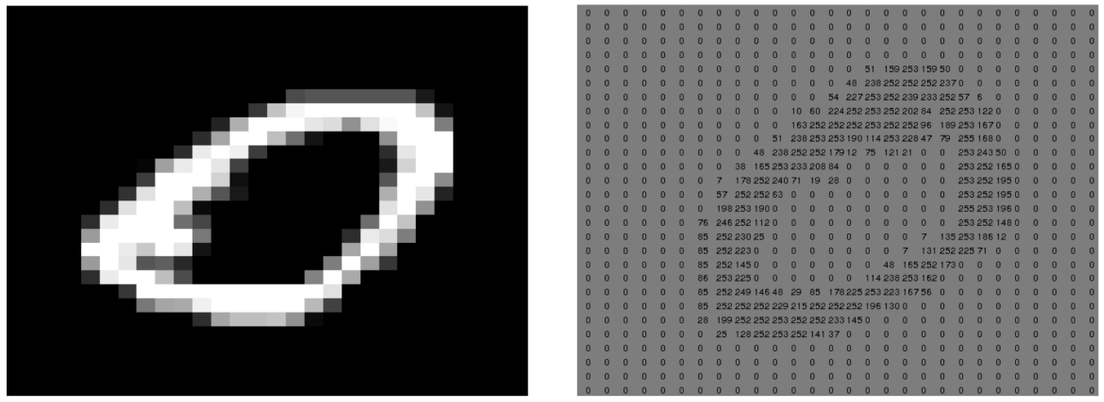


image —> vector

Including but not limited to:

(ref Deep Learning Book Chapter12)

- Computer Vision (example on CNN)
- Speech Recognition
- Natural Language Processing (example on RNN)

…

Ref:
[1] A Tutorial on Deep Learning, Quoc V. Le
[2] Deep Learning Book Chapter12 Applications

Choose a Framework

	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

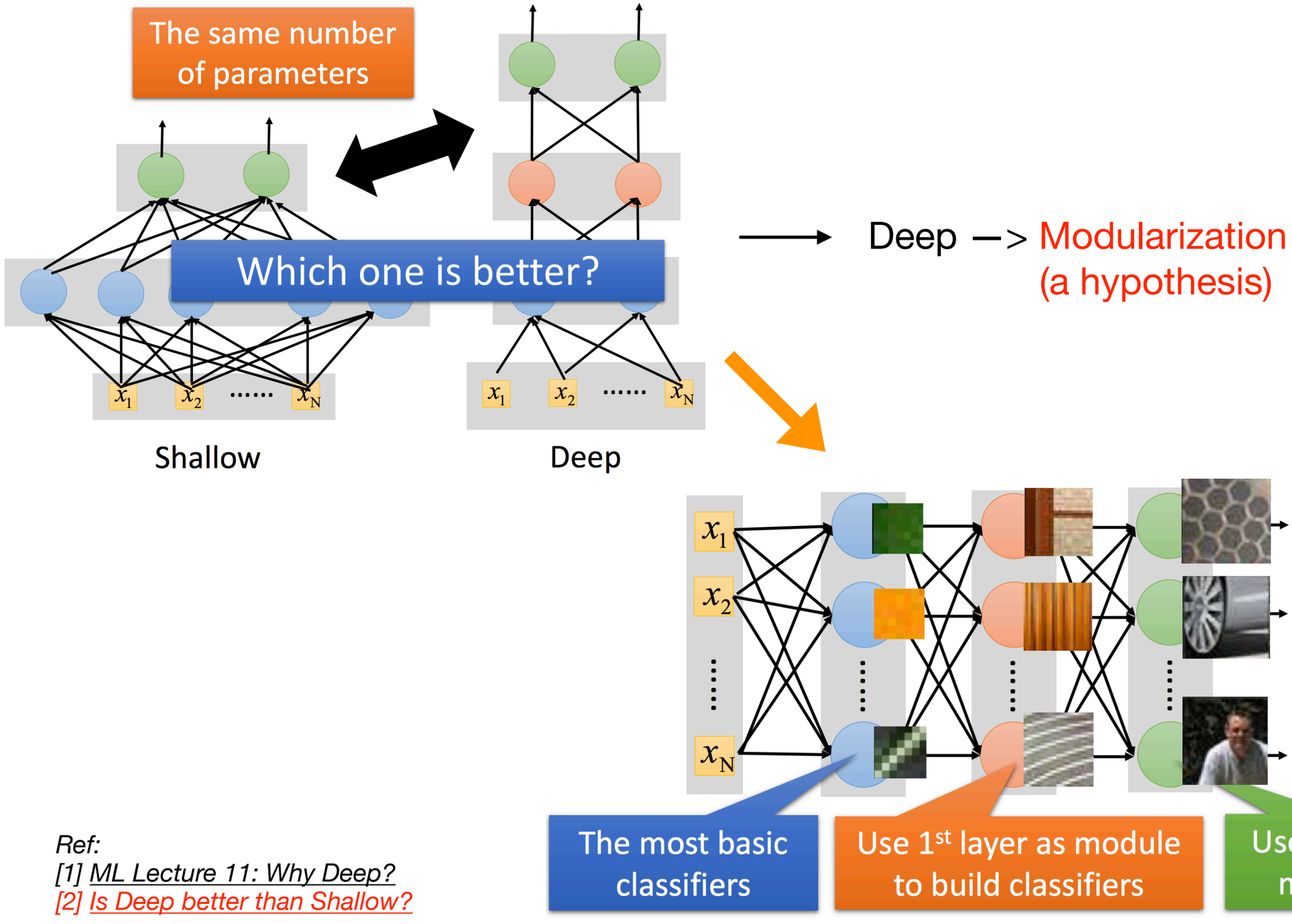
Hints:

- Most based on C++ or Python (mainstream, highly recommended).
- GPU is much faster than CPU (two versions).
- Just a tool, Theory (idea) is more important (implement from scratch).
- Keras is my recommended start (30 seconds to Keras).

Ref:

[1] 从*TensorFlow*到*Theano*：横向对比七大深度学习框架

Why Deep is better than Shallow?

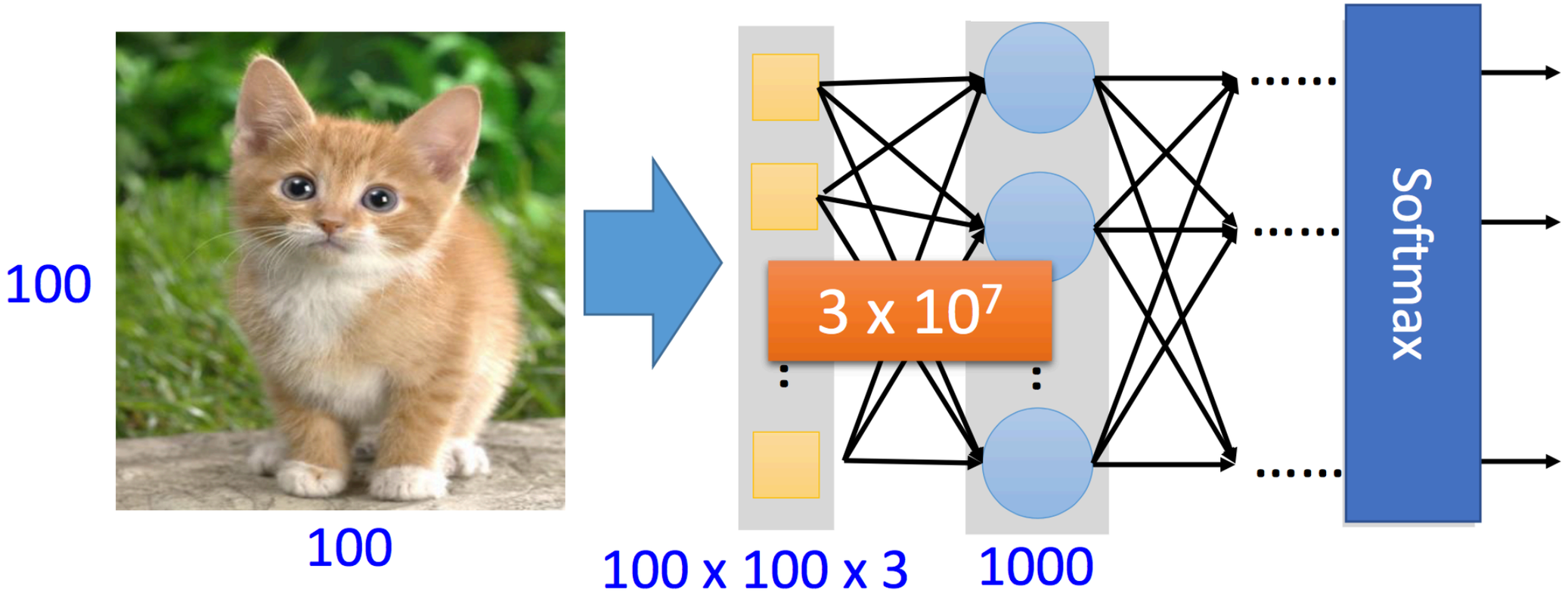


Ref:
[1] ML Lecture 11: Why Deep?
[2] Is Deep better than Shallow?

Content

- Introduction
- Special Networks
  - CNN
    - Motivation
    - Convolutional
    - Case (LeNet)
    - Implement
  - RNN
- Application
  - SST Prediction

Why CNN?

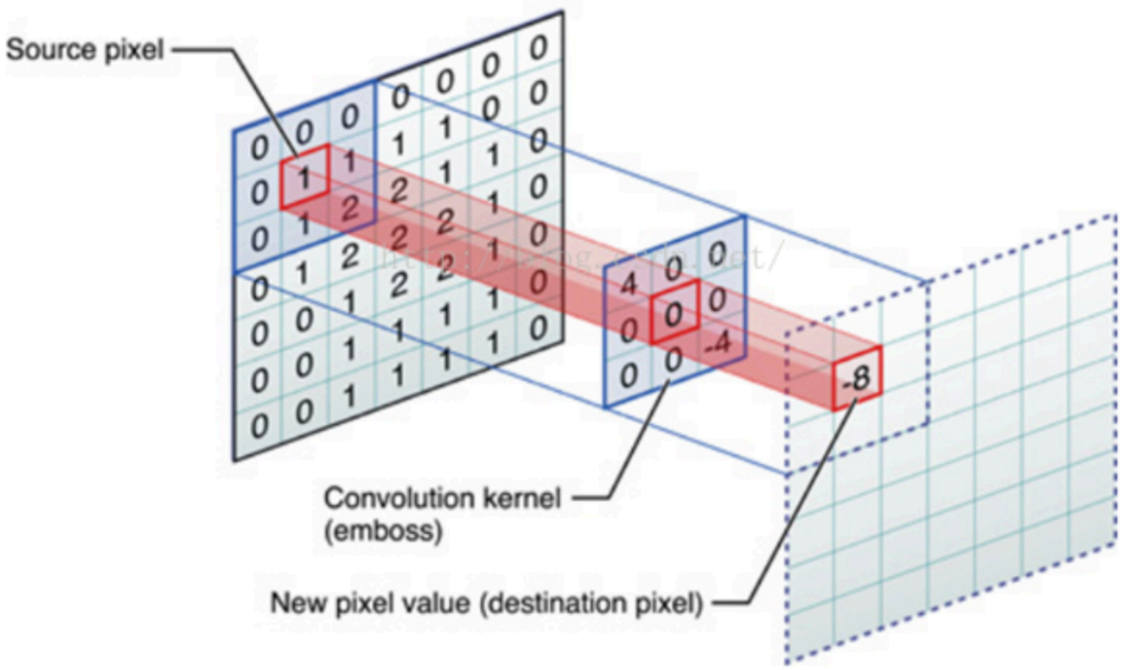


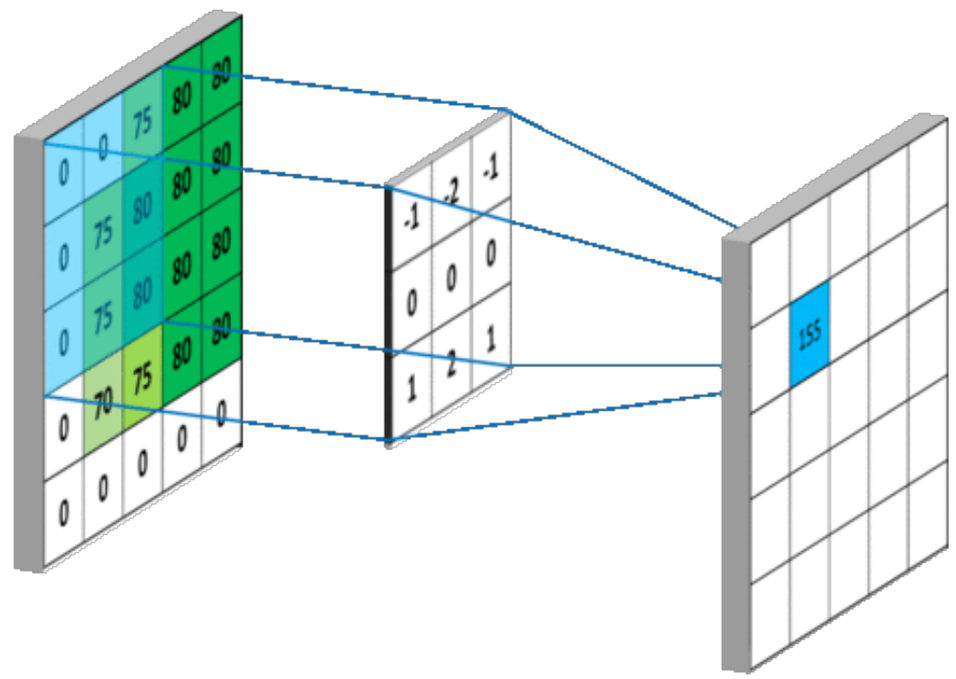
Problems: ***so many parameters to train*** *and* ***overfitting*** *problem.*

Two main ideas in CNN:

- Sparse (Local) Connectivity
  - Each neural only connects to part of the output of the previous layer, not fully connected, use **filter (kernel)**.
- Shared Weights (Parameter Sharing)
  - The neurons with different **receptive fields** can use the same set of parameters.

Convolutional





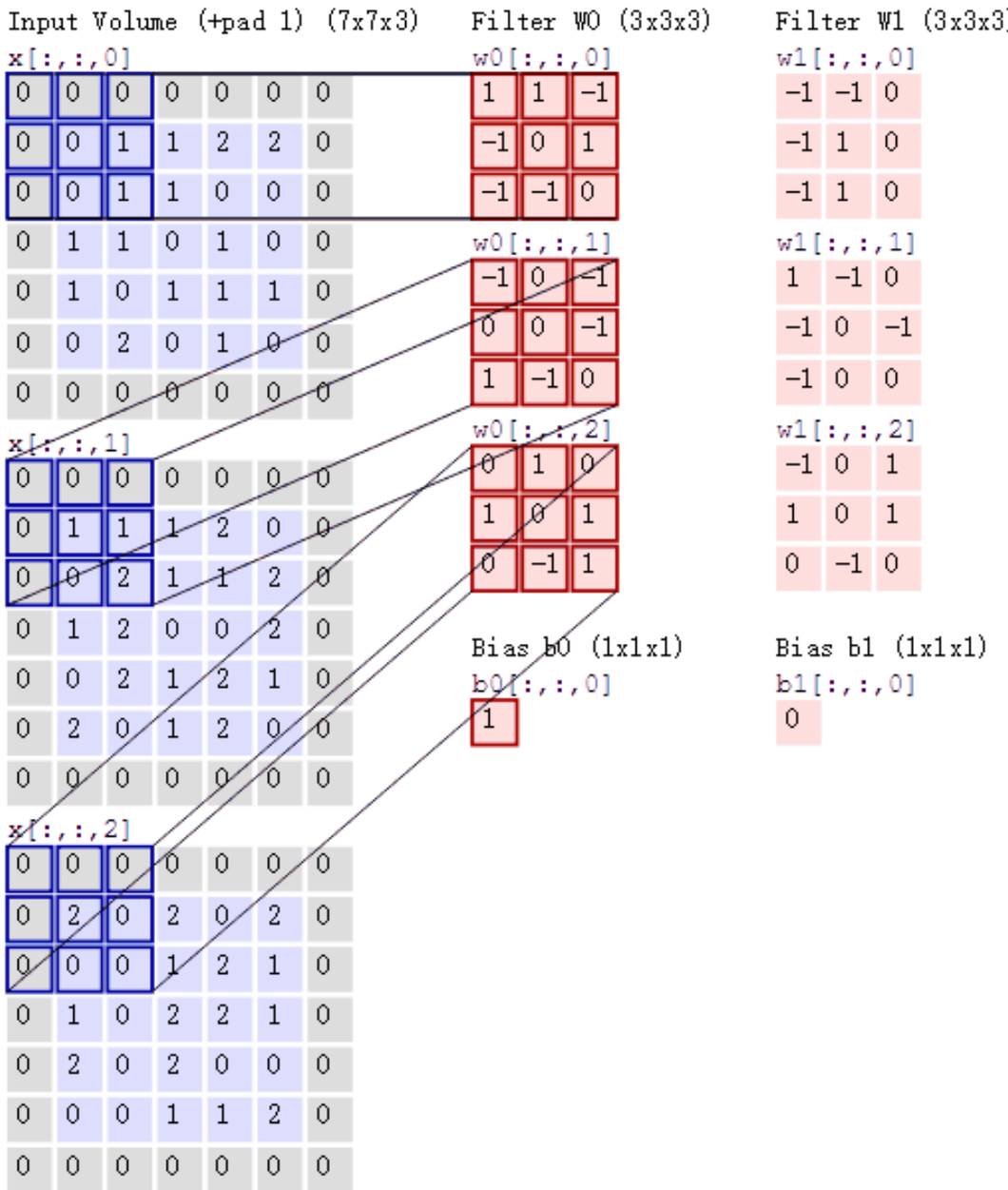
source Image * filter(kernel) —> new data matrix(feature map)

different Features of image (color, outline …)

Ref:

*[1] CNN*笔记：通俗理解卷积神经网络

Convolutional



Supplement:

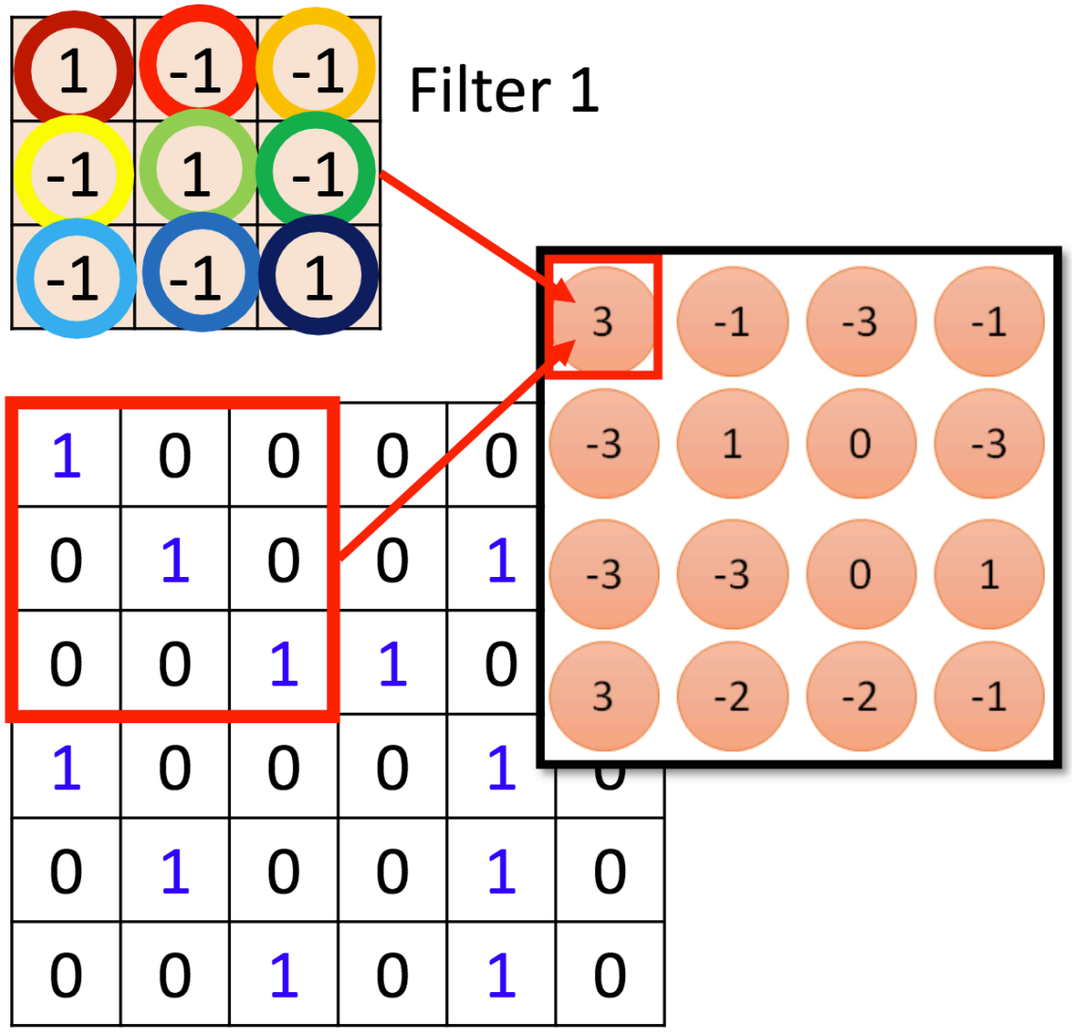
- RGB 3 channels
- two filters w0, w1 (3*3*3)
- stride=2 (each time move 2 step)
- zero-padding=1(full with 0)
- the output is called **feature map**

"Just think filter as your eyes, you can only focus on something (local) at a specified period of time."

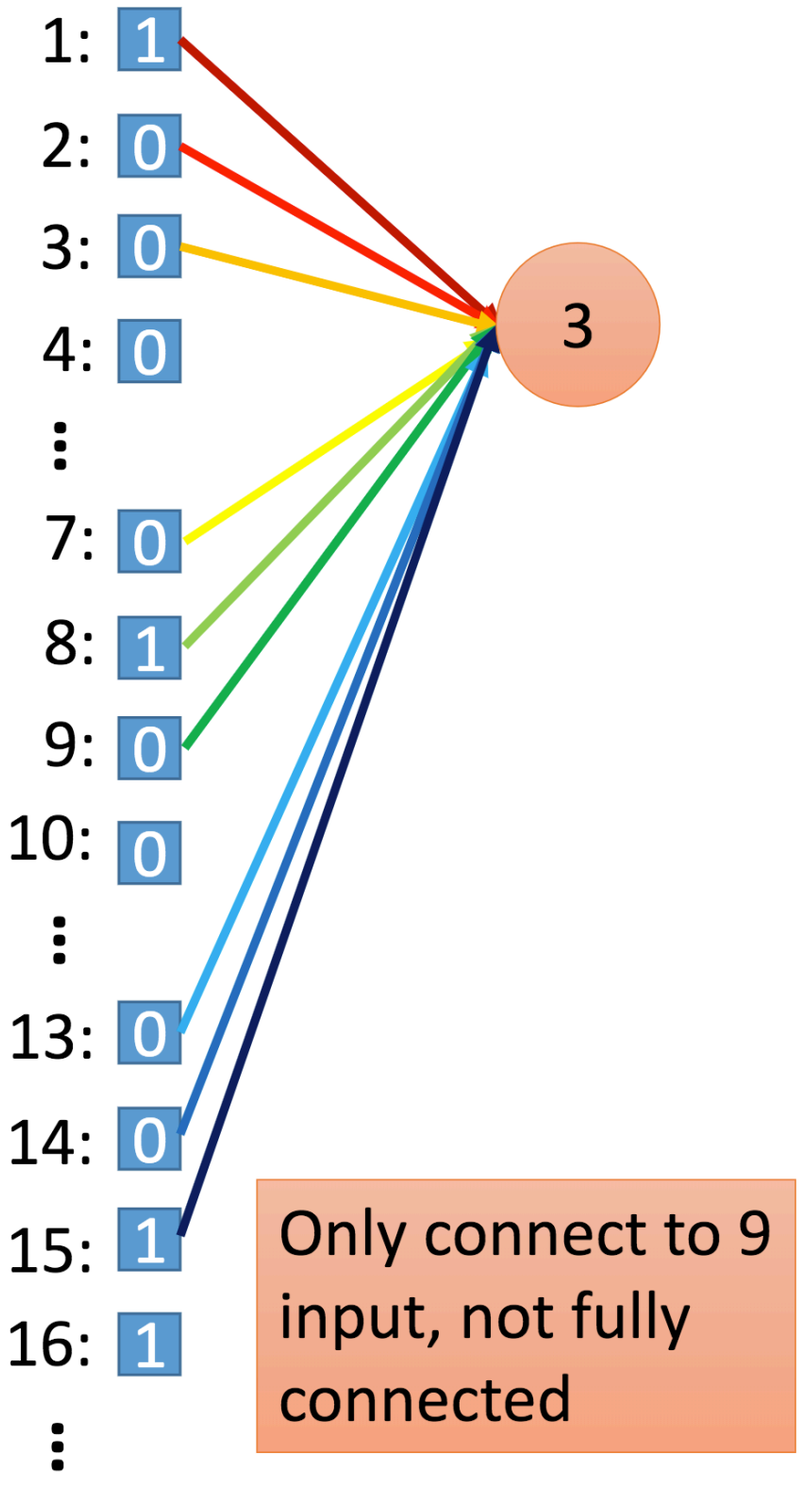
Ref:

[1] CS231n Convolutional Neural Networks for Visual Recognition Stanford University

CNN vs FNN (Sparse Connectivity)

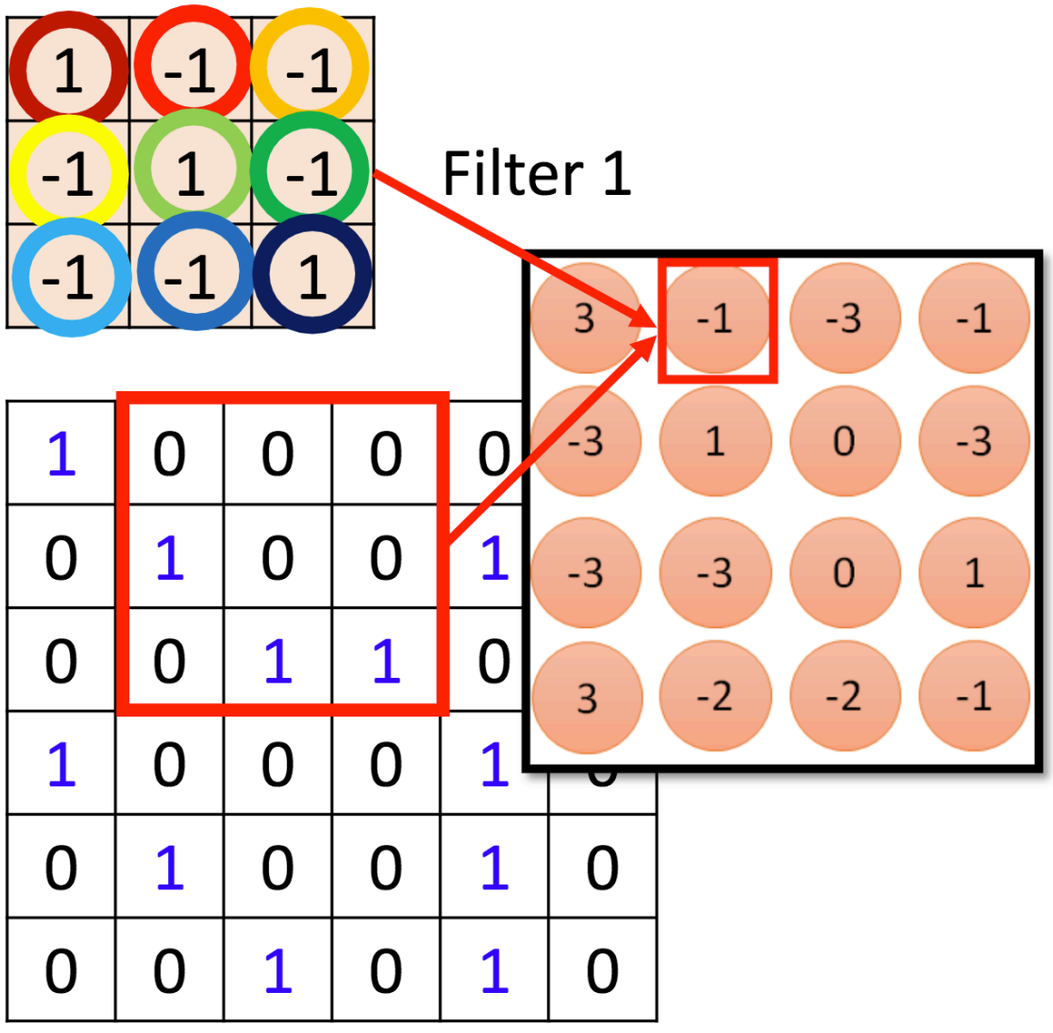


Less parameters!



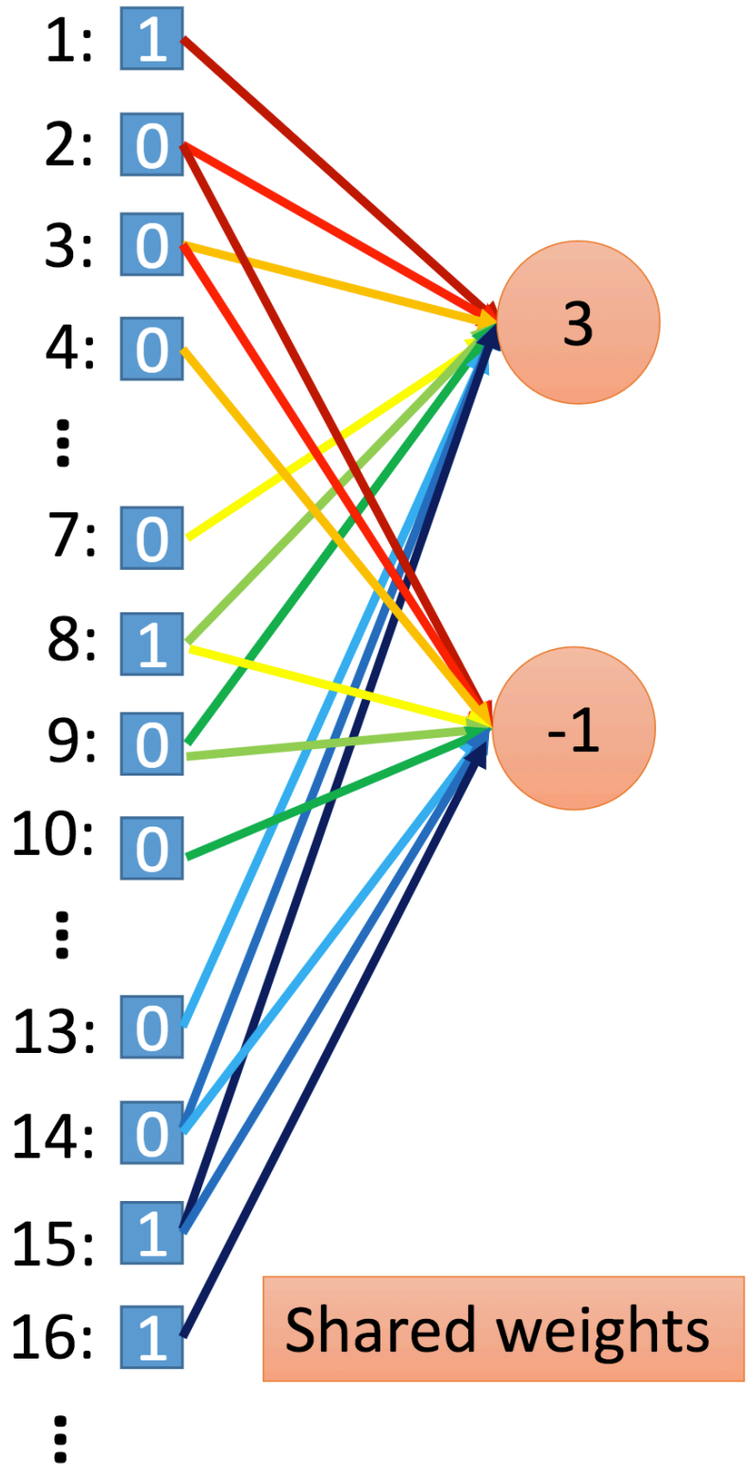
Only connect to 9 input, not fully connected

CNN vs FNN (Shared Weights)

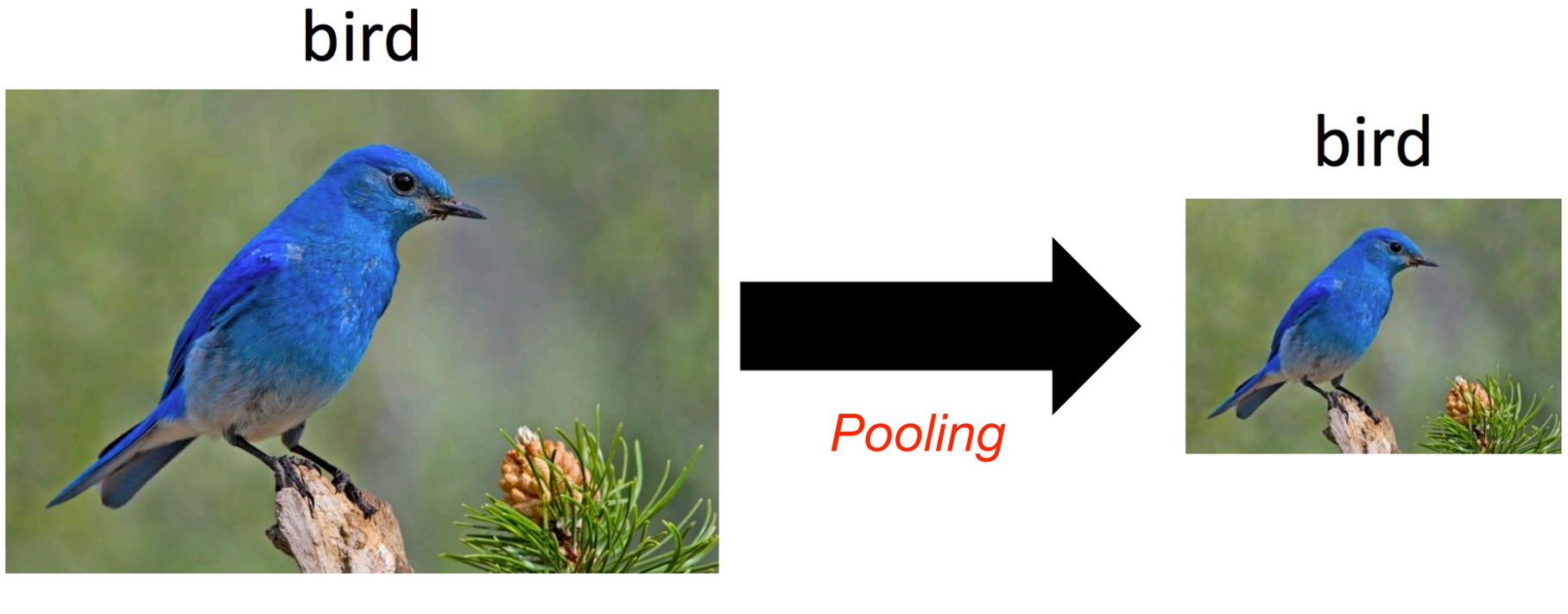


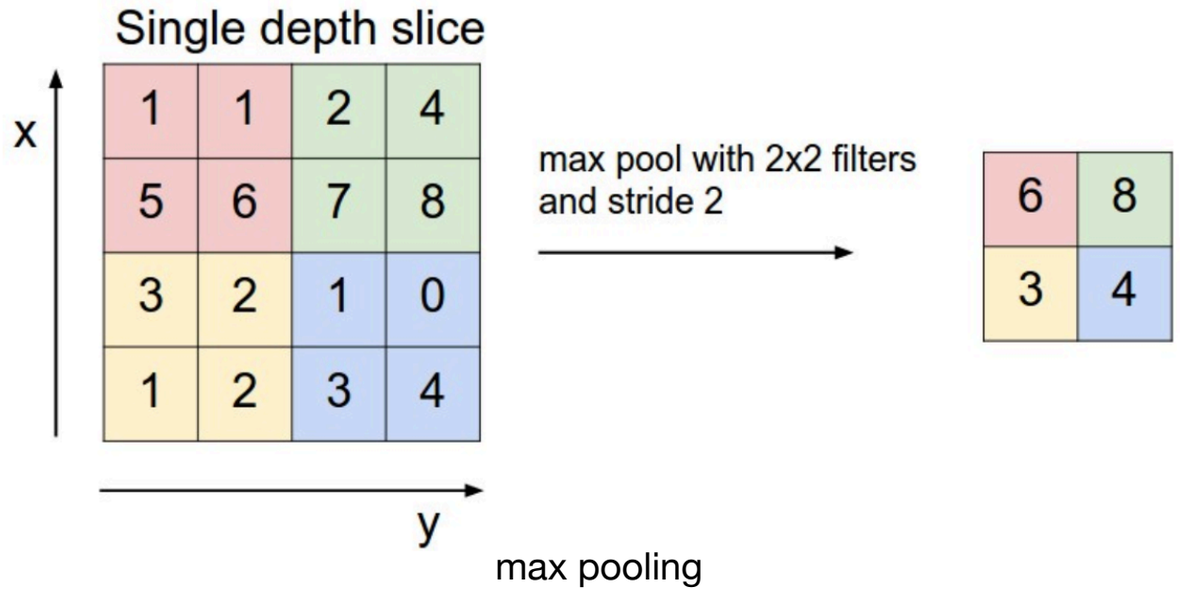
Less parameters!

Even less parameters!



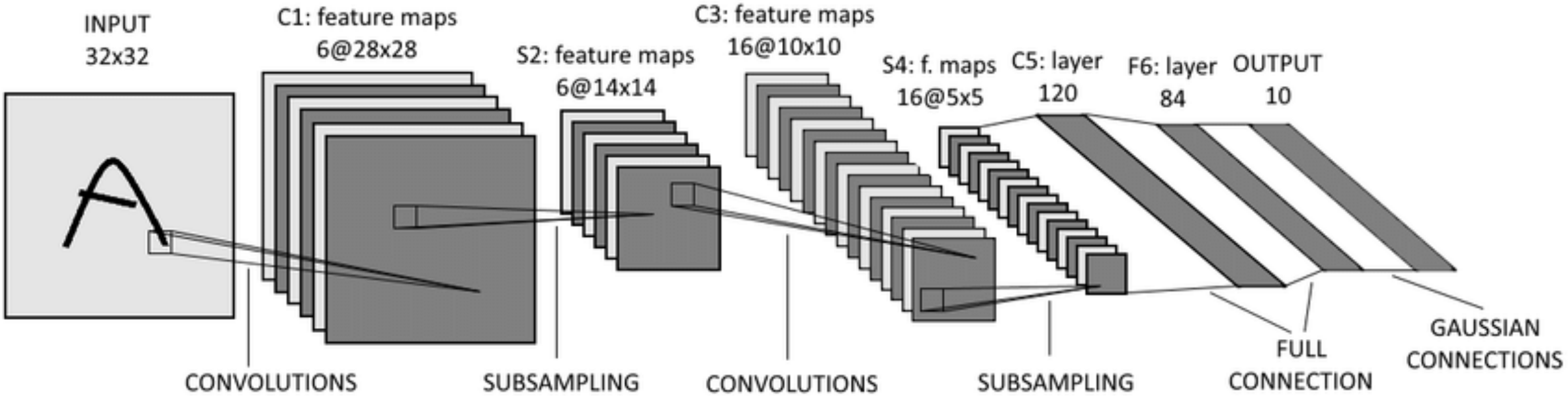
Pooling





Pooling (Subsampling):
a strategy to reduce the parameters:
Max-Pooling —> pick the maximum
Mean-Pooling —> pick the average

Case: LeNet



The first successful applications of Convolutional Networks

C1 is a convolutional layer, 6 filters with 5*5, feature map —> 28*28;
S2 is a pooling layer, 2*2 —> 1, so the size of data is reduced to 1/4, feature map —> 14*14;
C3 is a convolutional layer, 16 filters with 5*5, feature map —> 10*10;
S4 is a pooling layer, like S2, feature map —> 5*5;
C5 is a convolutional layer, 120 filters with 5*5, feature map —> 1*1;
F6 is a fully connected layer;

Ref:
[1] Gradient-Based Learning Applied to Document Recognition Y LeCun et al, 1998
[2] 卷积神经网络*(CNN)*学习笔记*1*：基础入门

Implement: CNN for MNIST Classification

- *Dataset:*

MNIST Database Classification
4 parts:
60000 training examples and labels (0~9)
10000 test examples and labels
all with fixed-size image (28*28)

- *Result:*

```
step 0, training accuracy 0.02
step 100, training accuracy 0.88
step 200, training accuracy 0.92
step 300, training accuracy 0.98
step 400, training accuracy 0.98
step 500, training accuracy 0.96
step 600, training accuracy 0.96
step 700, training accuracy 0.96
step 800, training accuracy 0.92
step 900, training accuracy 0.98
Test accuracy 0.9641
Run: 120.58326077461243 s
```

- *Framework:*

TensorFlow
(you can choose any framework you like for this.)

Ref:

[1] TensorFlow学习笔记2：构建CNN模型

Details

Open a TensorFlow Session

```
import tensorflow as tf
sess = tf.InteractiveSession()
```

Build CNN model

```
# 设置卷积层和池化层
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides = [1, 1, 1, 1], padding = 'SAME')

def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'SAME')
```

Dropout

```
# dropout 训练时随机丢弃一部分节点数据来减轻过拟合
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

Set the complete model and parameters

```
    train_accuacy = accuracy.eval(feed_dict = {x: batch[0], y_: batch[1], keep_prob: 1.0})
    print("step %d, training accuracy %g" %(i, train_accuacy))
train_step.run(feed_dict = {x: batch[0], y_: batch[1], keep_prob: 0.5})
```

Further

- What does CNN learn?
  https://www.youtube.com/watch?v=FrKWiRv254g

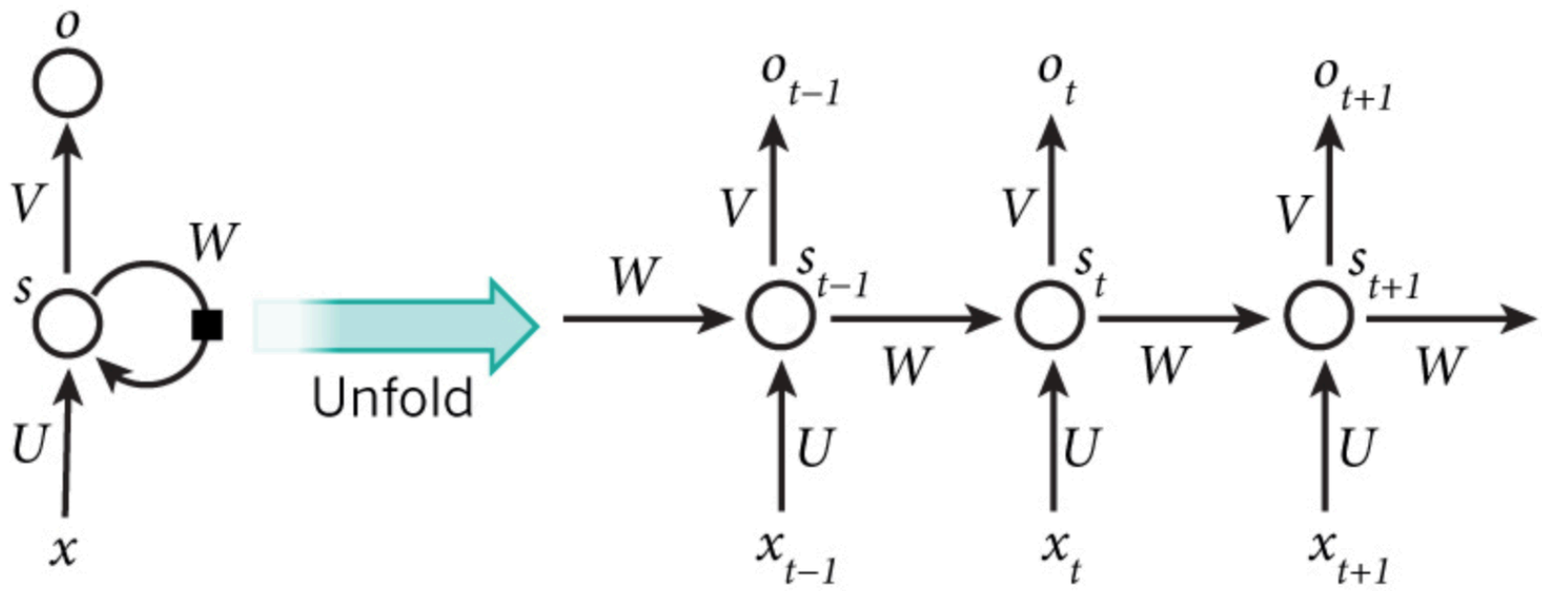
- What is wrong with convolutional neural nets?
  by Geoffrey Hinton
  https://www.youtube.com/watch?v=rTawFwUvnLE

- *Deep convolutional models: case studies*
  by Andrew Ng
  https://www.youtube.com/watch?v=dZVkygnKh1M

Content

- Introduction
- Special Networks
  - CNN
  - RNN
    - Motivation
    - LSTM Model
    - Implement
- Application
  - SST Prediction

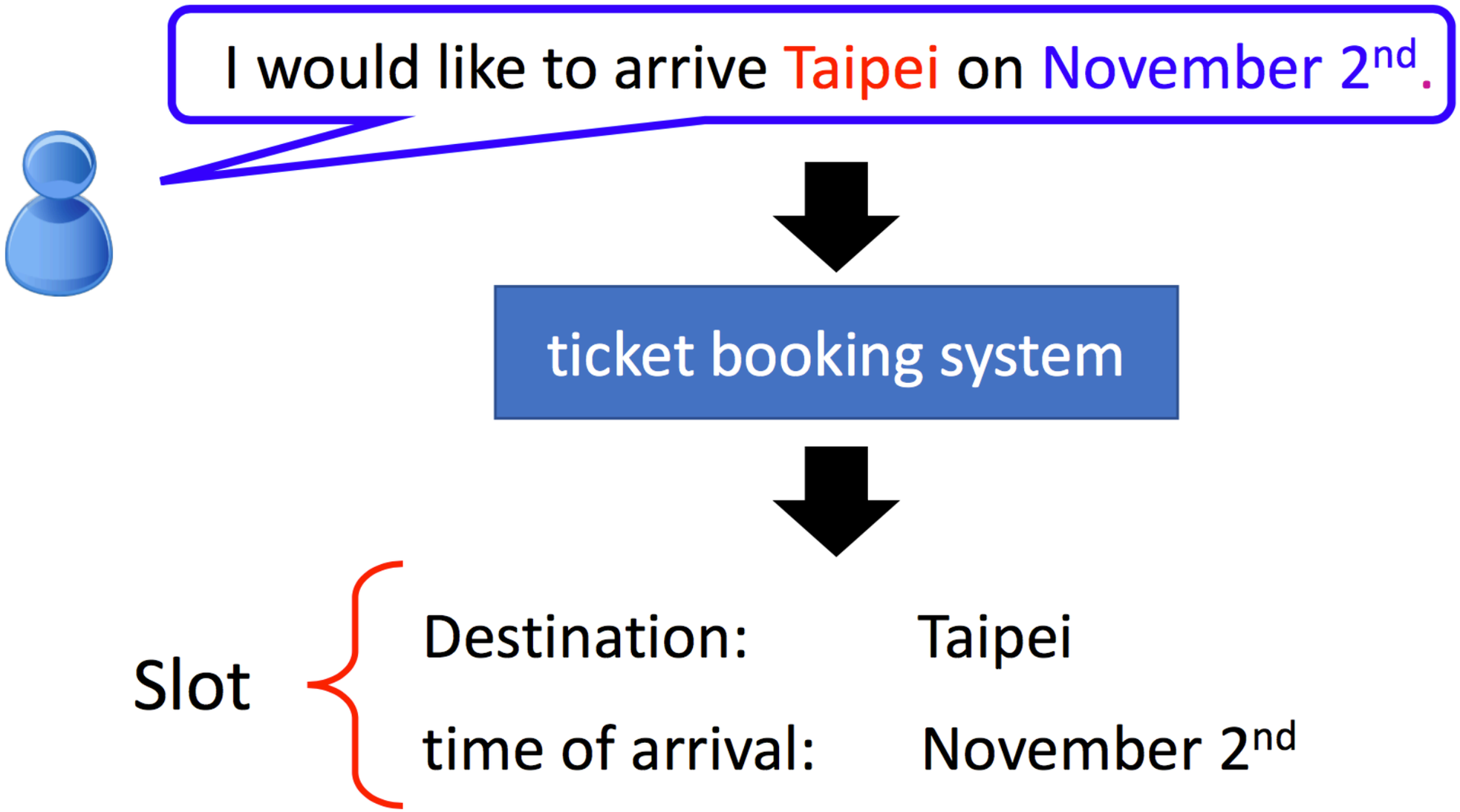
RNN(Recurrent Neural Network)



*"For tasks that involve **sequential** inputs, such as speech and language, it is often better to use RNNs. RNNs process an input sequence one element at a time, maintaining in their hidden units a '**state vector**' that implicitly contains information about **the history of all the past elements** of the sequence. "*

— 《Deep learning》 Yan LeCun et al. 2015 Nature

Example — Slot Filling



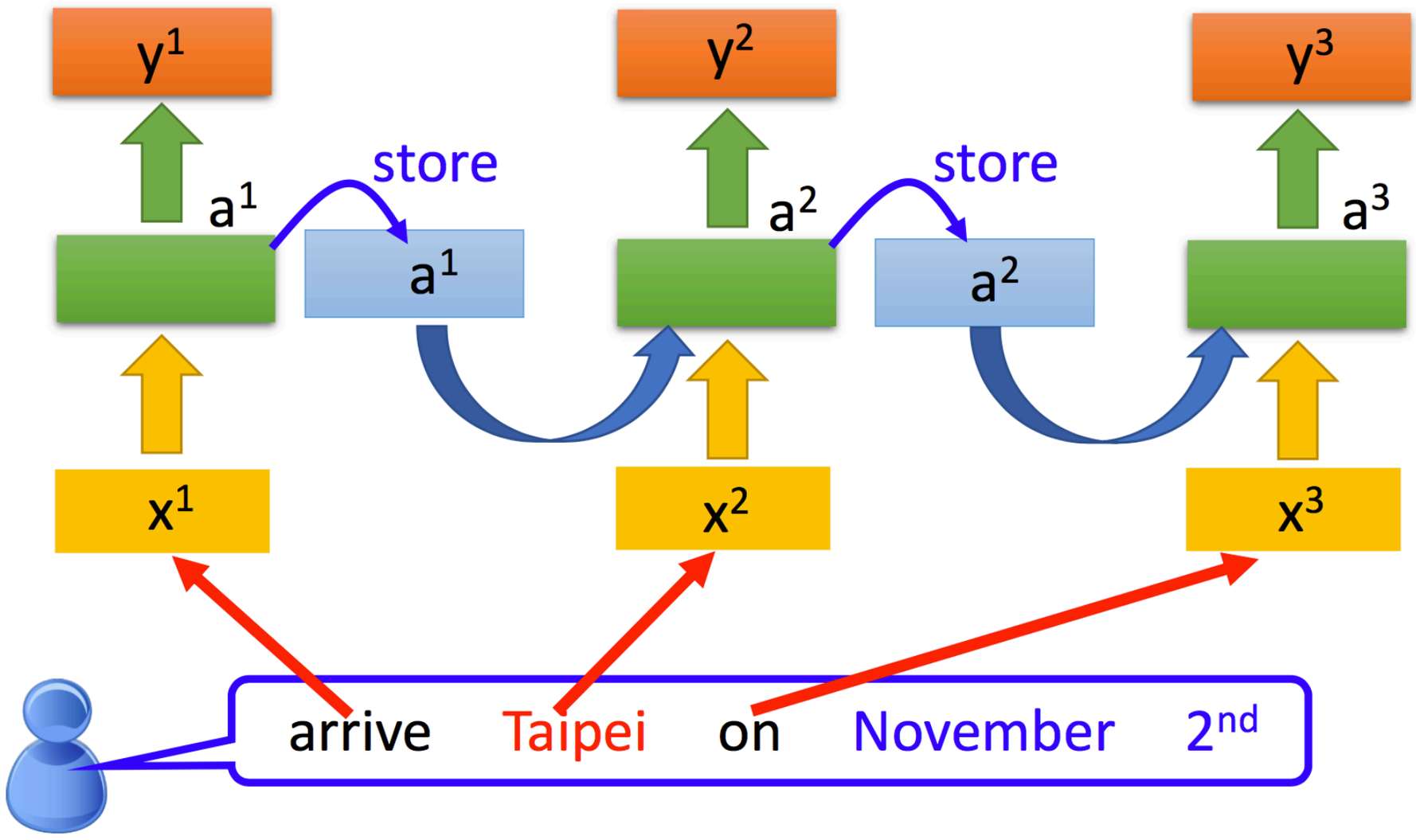
Ref:
[1] ML Lecture 25: Recurrent Neural Network (Part I)

Example — Slot Filling

Probability of "arrive" in each slot

Probability of "Taipei" in each slot

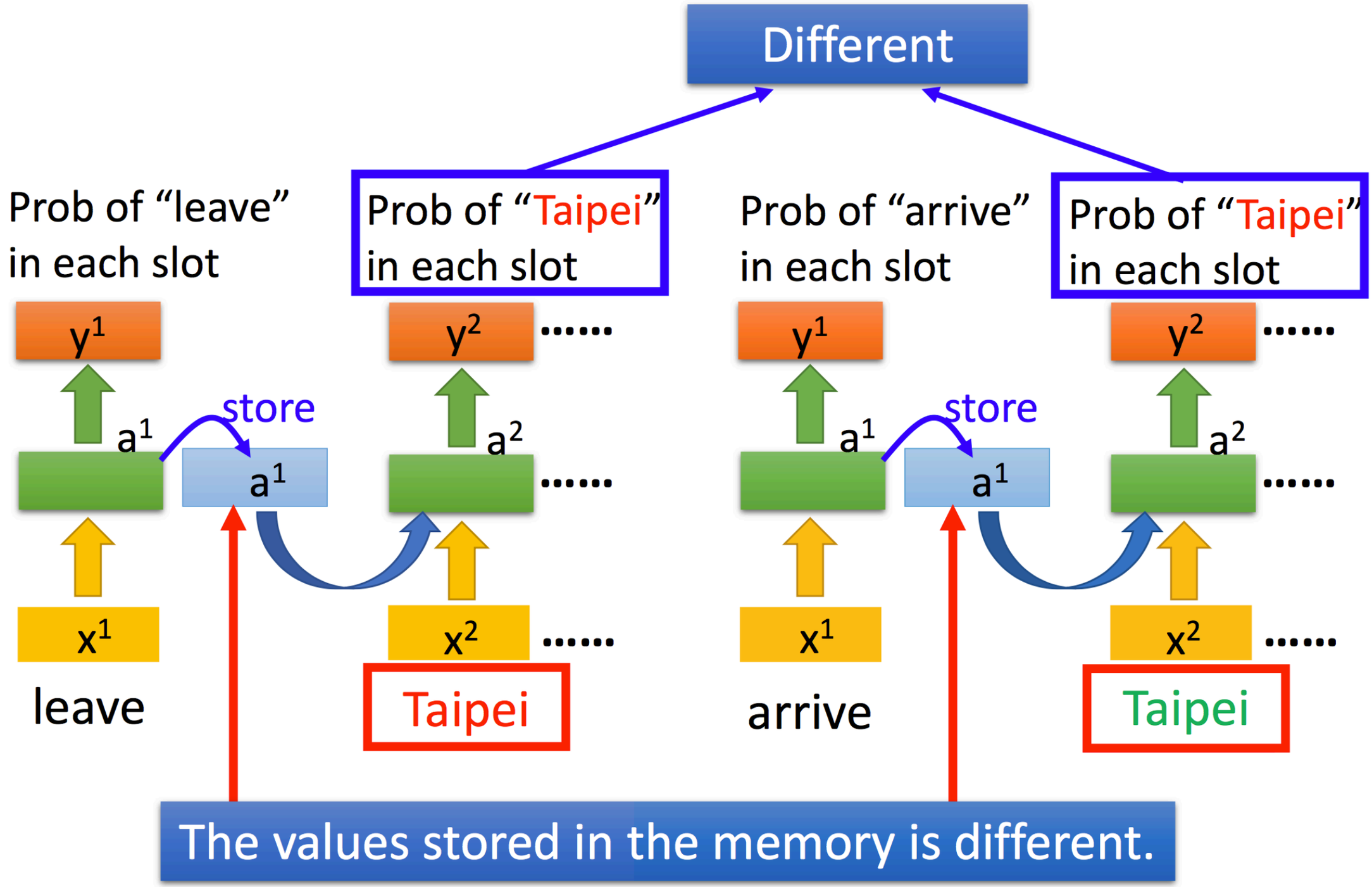
Probability of "on" in each slot



The same network reused many times!

Ref:
[1] ML Lecture 25: Recurrent Neural Network (Part I)

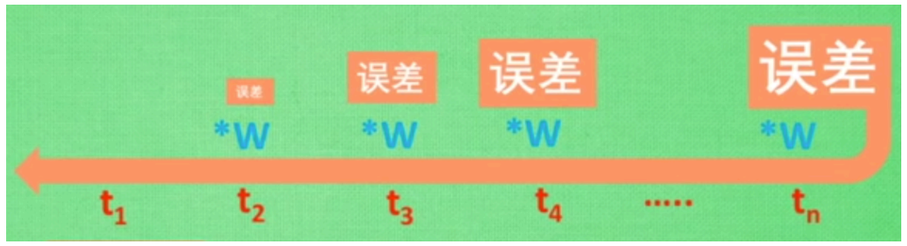
Example

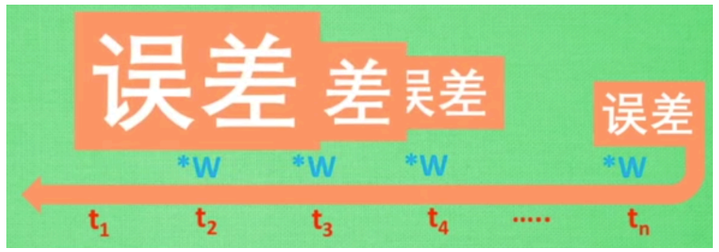

Different
Prob of "leave" in each slot
Prob of "Taipei" in each slot
Prob of "arrive" in each slot
Prob of "Taipei" in each slot
y^1
y^2
.....
y^1
y^2
.....
a^1
store
a^1
a^2
.....
a^1
store
a^1
a^2
.....
x^1
x^2
.....
x^1
x^2
.....
leave
Taipei
arrive
Taipei
The values stored in the memory is different.

What's the problem?

"I grew up in *China*, I am … …(skip ten thousand words) and I speak fluent *Mandarin*."

The gap between long-term dependency is so large!
Gradient Vanishing (w<1) and Gradient Exploding (w>1).





LSTM comes to rescue this!

Ref:
[1] 什么是 LSTM RNN 循环神经网络 (深度学习)? What is LSTM in RNN (deep learning)?
[2] Why it is difficult to train an RNN? Hinton on Coursera

More detail

$w = 1 \rightarrow y^{1000} = 1$

$w = 1.01 \rightarrow y^{1000} \approx 20000$

Large $\partial L/\partial w$ → Small Learning rate?

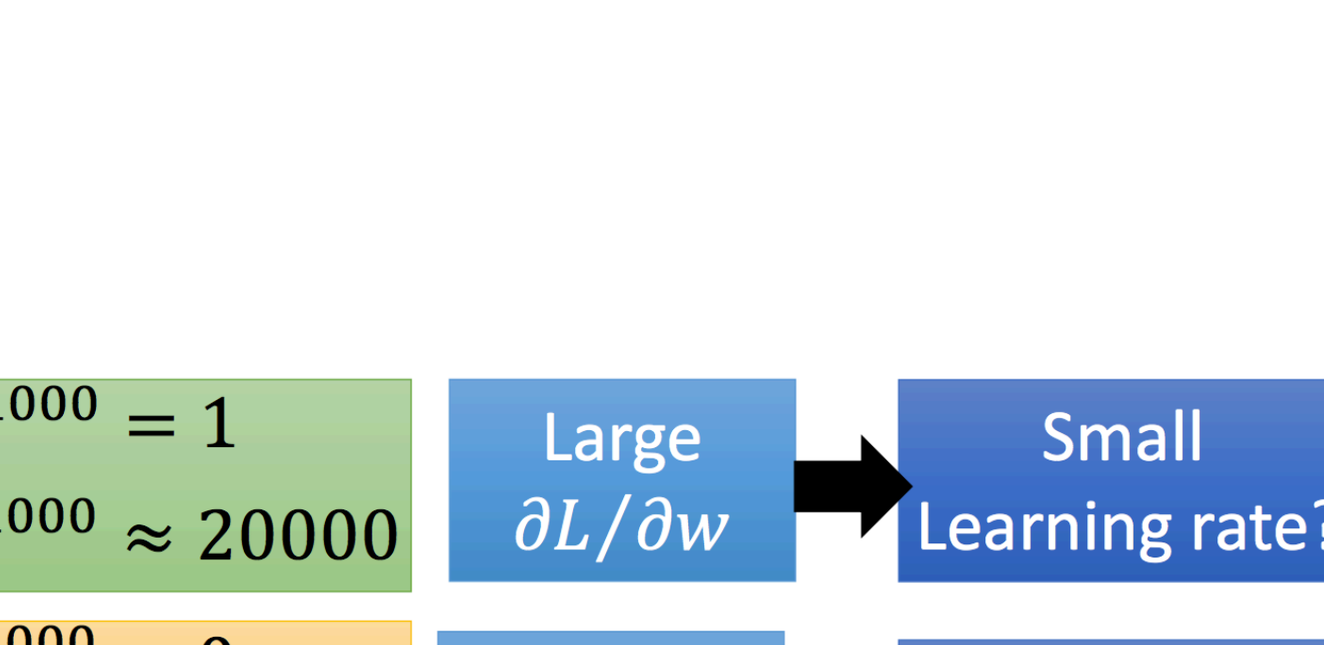
$w = 0.99 \rightarrow y^{1000} \approx 0$

$w = 0.01 \rightarrow y^{1000} \approx 0$

small $\partial L/\partial w$ → Large Learning rate?

Just think a single neural

We use **BPTT** (Backpropagation through time) to train RNN.

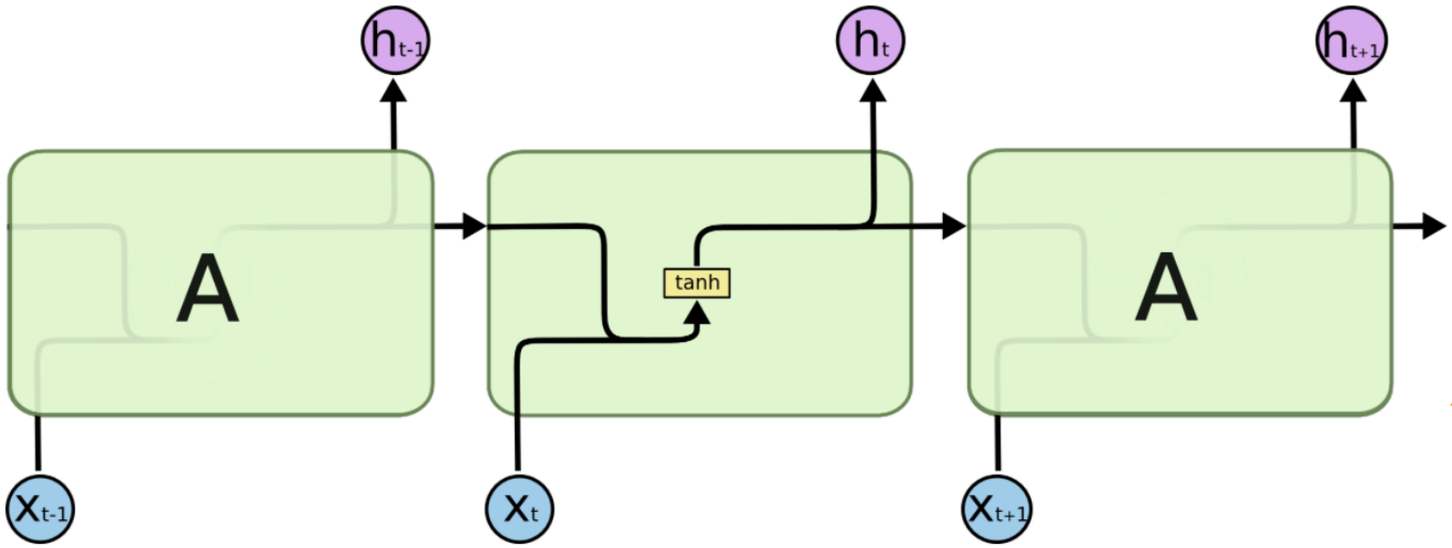


Ref:

[1] ML Lecture 26: Recurrent Neural Network (Part II)

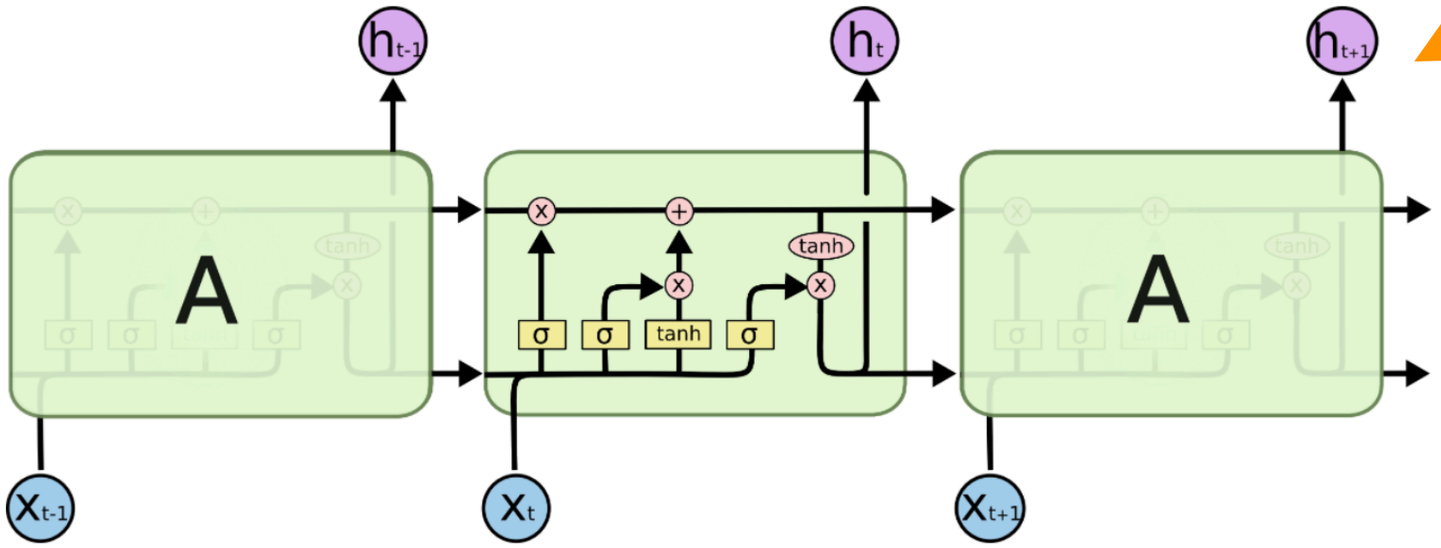
[2] Learning Long-Term Dependencies with Gradient Descent is Difficult Y. Bengio

LSTM(Long Short-Term Memory)



standard RNN

a single neural layer
—>
four neural layers

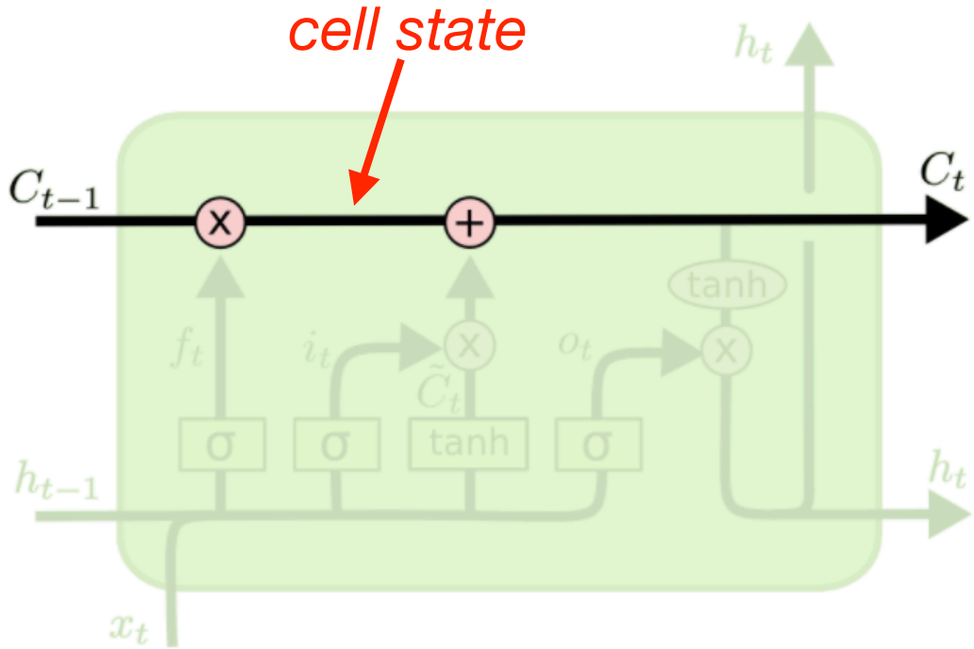


LSTM

Ref:
[1] Understanding LSTM Networks
[2] 零基础入门深度学习(6) - 长短时记忆网络(LSTM)
[3] Long Short Term Memory Hochreiter & Schmidhuber 1997

LSTM

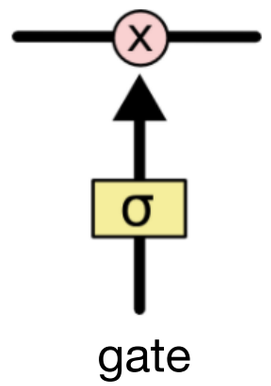
"The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates."



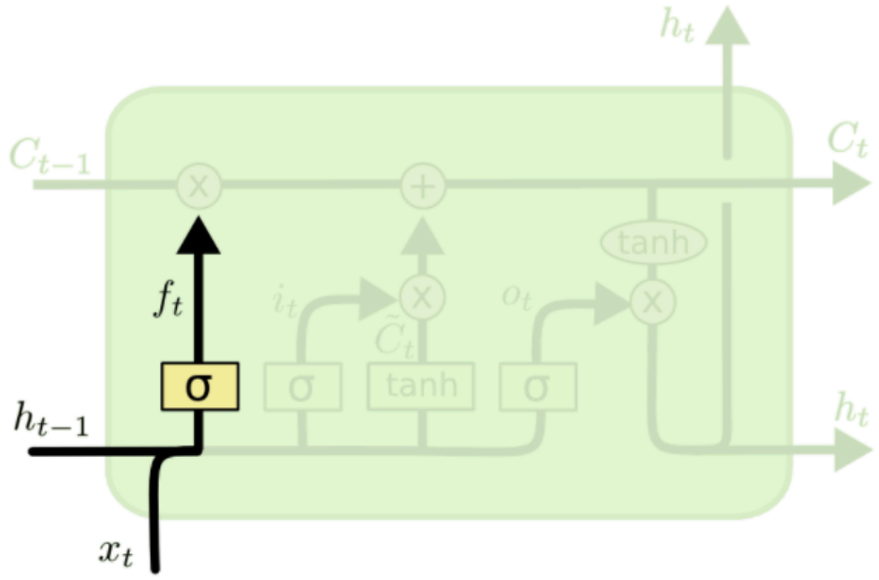
"Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation."

Three kind of gate in LSTM:

- Input gate
- Output gate
- Forget gate



What to forget



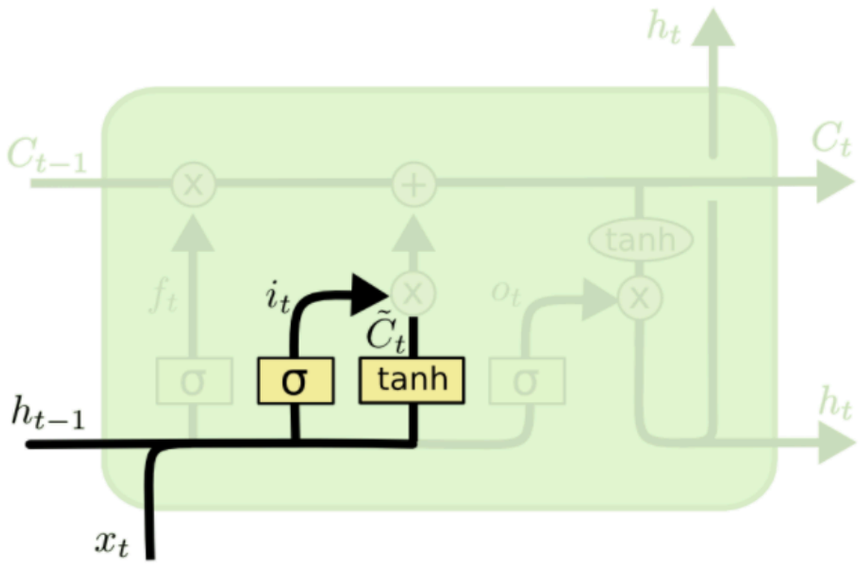
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

last output *current input*

sigmoid function *bias*

First step: to decide what information throw away from the cell state.
Use Forget gate layer, accept h_{t-1} and x_t ,output a number between 0~1
1 —> “completely keep this”
0 —> “completely get rid of this”

What to store



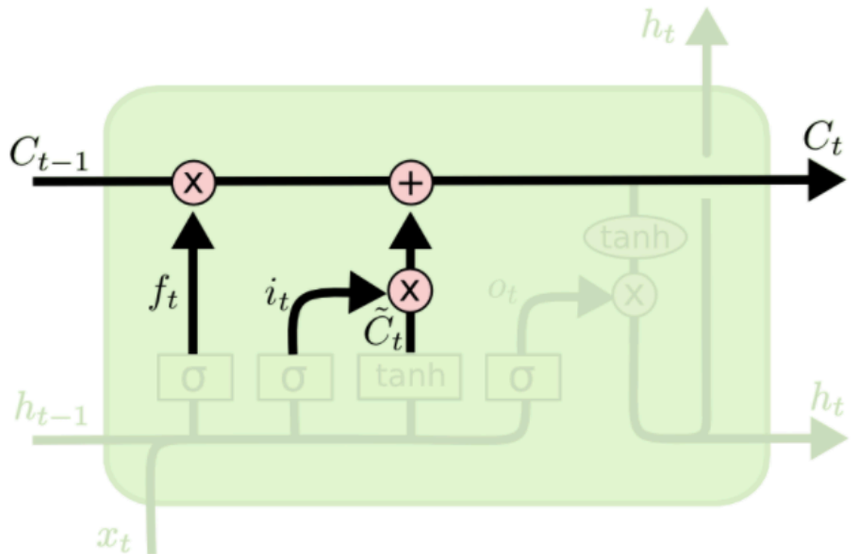
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Second step: to decide what information to store in the cell state.

- Use Input gate layer to decides which values to update
- Use a *tanh* layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.

Update cell state



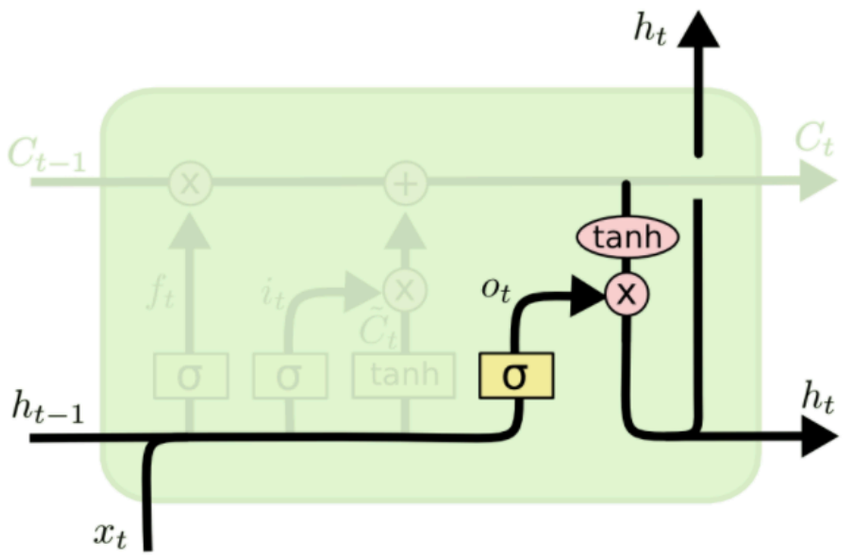
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hadamard Product

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 * 3 \\ 2 * 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

Update the old cell state, C_{t-1} —> C_t

Output



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

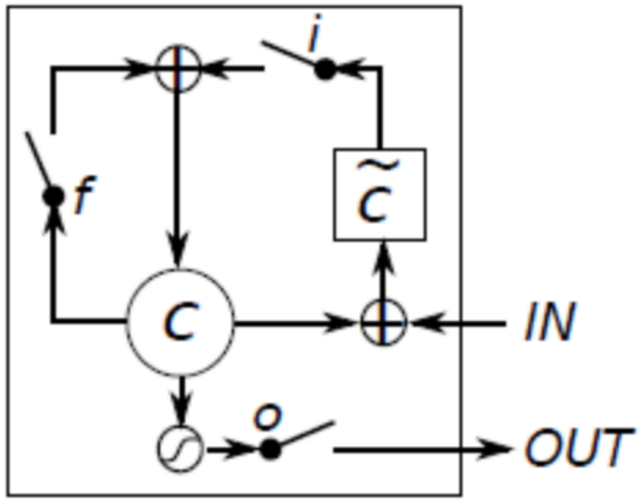
$$h_t = o_t * \tanh\left(C_t\right)$$

Output is based on cell state, but will be a filtered version.
Use Output gate layer.

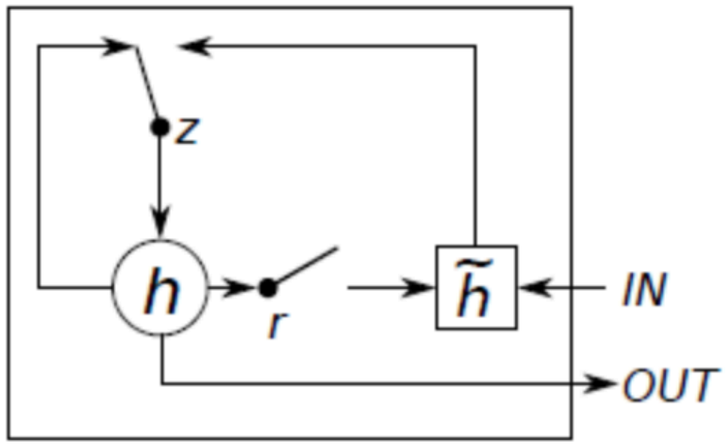
GRU

“In fact, it seems like almost every paper involving LSTMs uses a slightly different version. ”

— 《Understanding LSTM Networks》 Christopher Olah



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

GRU (Gated Recurrent Unit) —> Only two gates (reset gate and update gate)
Less parameters, Similar effect.

Implement: LSTM for Sequence Classification

- *Dataset:*
IMDB movie sentiment classification
50000 reviews (good / bad)
50% Training 50% Test

- *Framework:*
Keras (Using TensorFlow backend)

- *Result:*

Single LSTM

```
Epoch 1/3
2017-12-16 15:46:54.421209: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU suppor
ts instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
25000/25000 [==============================] - 206s 8ms/step - loss: 0.4695 - acc: 0.7779
Epoch 2/3
25000/25000 [==============================] - 204s 8ms/step - loss: 0.3023 - acc: 0.8760
Epoch 3/3
25000/25000 [==============================] - 203s 8ms/step - loss: 0.2509 - acc: 0.9011
Accuracy: 86.99%
```

3 epochs
Acc: 86.99%

Convolutional LSTM

```
25000/25000 [==============================] - 98s 4ms/step - loss: 0.4286 - acc: 0.7892
Epoch 2/3
25000/25000 [==============================] - 114s 5ms/step - loss: 0.2438 - acc: 0.9053
Epoch 3/3
25000/25000 [==============================] - 126s 5ms/step - loss: 0.2005 - acc: 0.9236
Accuracy: 88.40%
```

3 epochs
Acc: 88.40%

Ref:
[1] Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
[2] Learning Word Vectors for Sentiment Analysis Andrew L. Maas et. 2011

Content

- Introduction
- Special Networks
  - CNN
  - RNN
- Application
- SST Prediction
  - Problem
  - Model
  - Experiment
  - Conclusion
  - Insight

Paper

Prediction of Sea Surface Temperature Using Long Short-Term Memory

Qin Zhang, Hui Wang, Junyu Dong, Guoqiang Zhong, and Xin Sun
Department of Computer Science and Technology, Ocean University of China

IEEE Geoscience and Remote Sensing Letters (Volume: 14, Issue: 10, Oct. 2017)
Impact Factor: 2.761

Problem

SST Prediction in coastal seas —> a time series regression problem

Data(input): latitude + longitude + interval of time = 3-D grids (a sequence of real value)

Formulation: if k days' SST values are given, what are the SST values for the k + 1 to k + l days? Here, l represents the length of prediction.

Classical Method

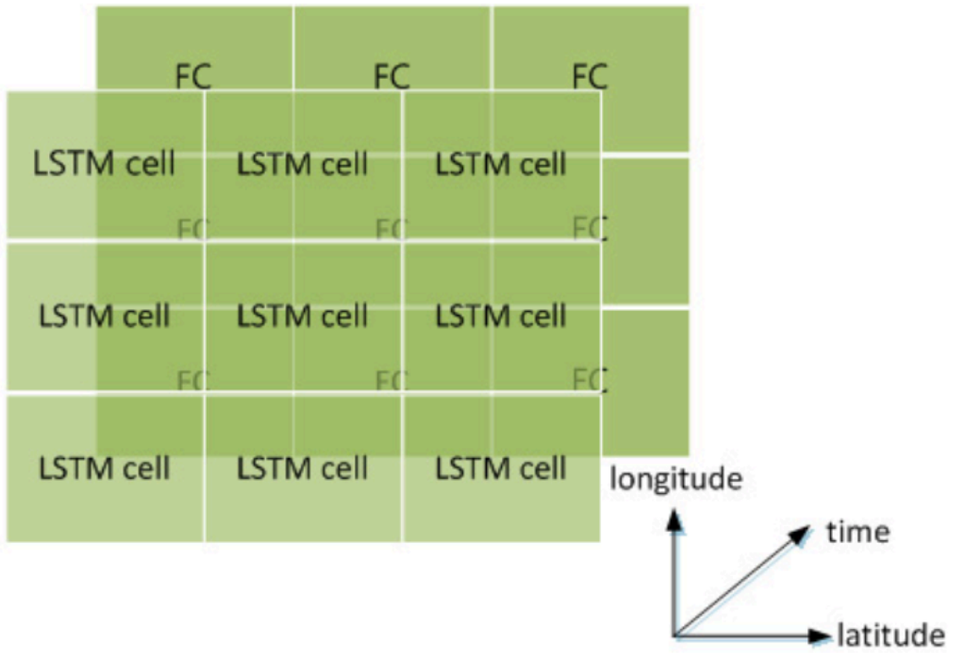
Numerical model

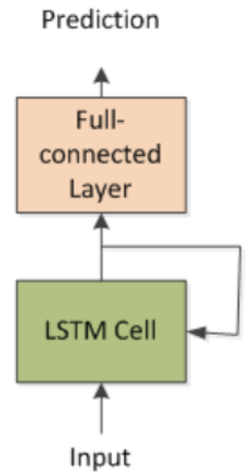
- Based on physics, utilize a series of differential equations to describe the variation of SST (sophisticated and resource-intensive)
- Differ in different sea areas (not universality)

Data-driven model *(this paper)*

- Based on data
- linear regression, SVM, neural network…

Model





$$(h_i, m_i) = \mathrm{LSTM}\left(\begin{bmatrix} \mathrm{Iinput} \\ h_{i-1} \end{bmatrix}, m, W\right)$$

$$\mathrm{prediction} = \sigma\left(W^{\mathrm{fc}} h_l + b^{\mathrm{fc}}\right)$$

current input

cell state

previous output

LSTM —> capture the temporal relationship, solve the long-range dependency.
FC (fully-connected) —> “make a better abstraction and combination for the output vector, and reduces its dimensionality”.

Experiment

Data: SST daily mean values

- Source: NOAA(https://www.esrl.noaa.gov/psd/)
- Time Interval: September 1981 to November 2016, 12868 days totally
- Resolution:0.25 * 0.25 (1440 * 720)
- Region: Bohai Sea (37.07N to 41N, 117.35E to 121.10E, 16 * 15)
- Partition:
    - Training: September 1981 to August 2012 (11323)
    - Validation: September 2012 to October 2012 (122)
    - Testing: January 2013 to December 2015 (1095)

Comparison: support vector regression (SVR)
multilayer perceptron regression (MLPR)

Metric: the root mean squared error (RMSE)

Implement: Keras for LSTM, Scikit-learn for SVR and MLPR

Experiment

TABLE I

PREDICTION RESULTS (RMSE) ON FIVE LOCATIONS WITH DIFFERENT *Units_r*s

$units\_r$	p_1	p_2	p_3	p_4	p_5
1	0.1595	0.1171	0.2690	0.2988	0.2626
2	0.1589	0.1137	0.2569	0.2909	0.2695
3	0.2075	0.0923	0.2580	0.2819	0.2606
4	0.2152	0.0918	0.2349	0.2752	0.2672
5	**0.1280**	**0.0914**	**0.2310**	0.2723	**0.2362**
6	0.1353	0.0922	0.2454	**0.2646**	0.2468

TABLE II

PREDICTION RESULTS (RMSE) ON FIVE LOCATIONS WITH DIFFERENT l_r s

l_r	p_1	p_2	p_3	p_4	p_5
1	**0.1280**	**0.0914**	**0.2310**	**0.2723**	**0.2362**
2	0.1288	0.1153	0.2500	0.2730	0.2496
3	0.3659	0.0950	0.2656	0.2732	0.3334

TABLE III

PREDICTION RESULTS (RMSE) ON FIVE LOCATIONS WITH DIFFERENT *k*s

l_{fc}	p_1	p_2	p_3	p_4	p_5
1[3]	**0.1280**	**0.0914**	**0.2310**	**0.2723**	**0.2362**
2[3,3]	0.2838	0.0945	0.2880	0.2724	0.2461
2[6,7]	0.3660	0.2605	0.4655	0.2730	0.3355

Parameters Setting:

- Single location SST prediction (5 random locations with 15 days length in the Bohai SST data)
- Different LSTM units (an empirical value range $\lfloor (l/2), 2l \rfloor$)
- Different LSTM layer (1 is better)
- Different FC units (same with prediction length *l*)
- Different FC layer (1 is better)

Experiment

TABLE IV

PREDICTION RESULTS (AREA AVERAGE RMSE) ON THE BOHAI SST DATA SET

Methods	Daily		Weekly	Monthly
	one day	three days	one week	one month
SVR	0.3998	0.6158	0.4716	0.6538
MLPR	0.6633	0.8215	0.6919	0.8360
LSTM network	**0.0767**	**0.1775**	**0.3844**	**0.3928**

Supplement:

- LSTM is better than SVR and MLPR in Daily, weekly and monthly.

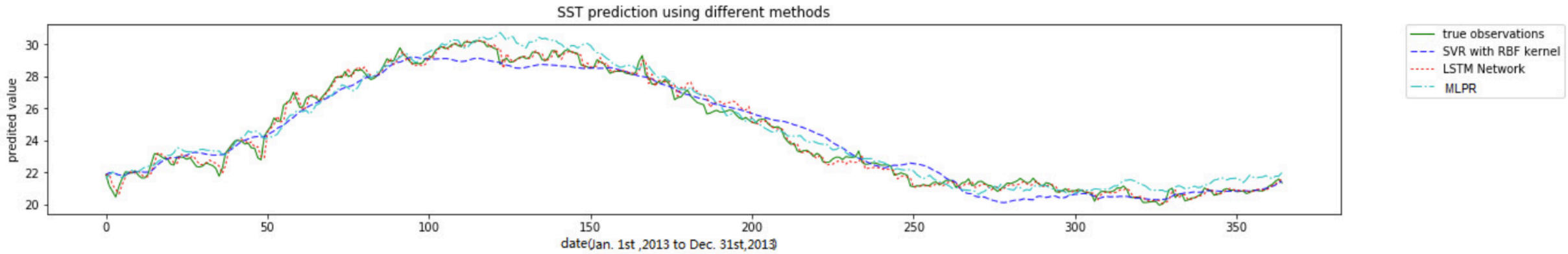


Fig. 5. SST Three-Days' Prediction at One Location Using Different Methods.

Experiment

TABLE VI

PREDICTION RESULTS(AREA AVERAGE RMSE) ON BOHAI SST DAILY DATA SET IN 2016

Model	Prediction of 2016	
	one day	three days
original	0.1346	0.2145
updated	**0.0899**	**0.1843**

Online Model(fine-tuning):

- with 2013, 2014, 2015 data as training data for predict 2016
- updated model performs the best, while other method cannot do this.

Conclusion

- Proposed a LSTM-based network for SST prediction and confirm its effectiveness (better than traditional methods).
- The online update characteristics of the proposed method are shown (fine-tuning) .
- There may not be enough training samples in our method (more data is needed).

Insight

- Single point —> Region (no much detail on this paper)
  (*“using the basic LSTM block to predict the SST for a single location. Then, we evaluate the proposed method on area SST prediction for Bohai Sea.”*)

- ENSO is a much complex case and more meaningful.
  (not single regression problem, occurence, develop and type …)

- The structure of the network is critical, need more experiments for that.
  (ongoing)