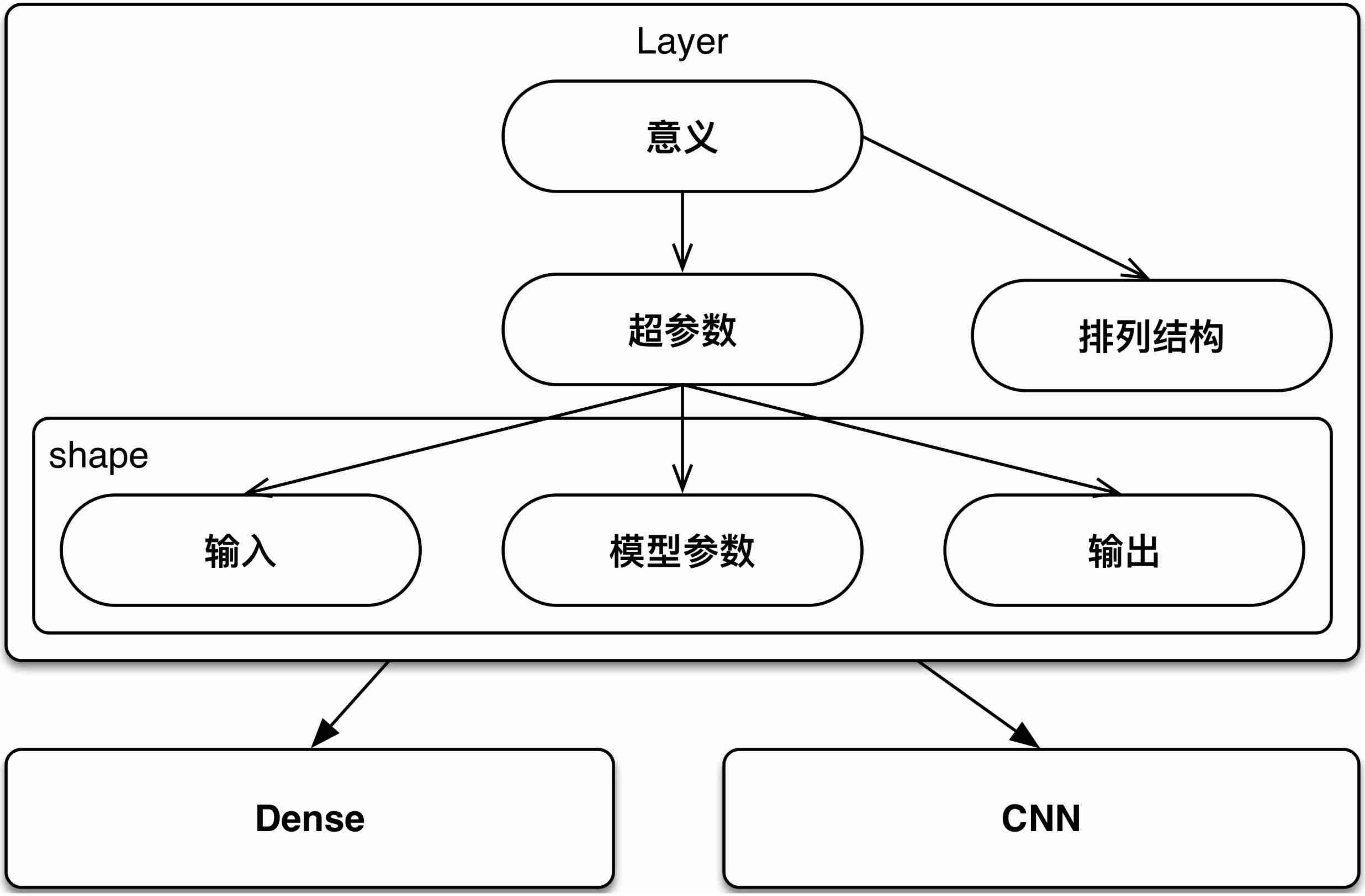
CNN & RNN

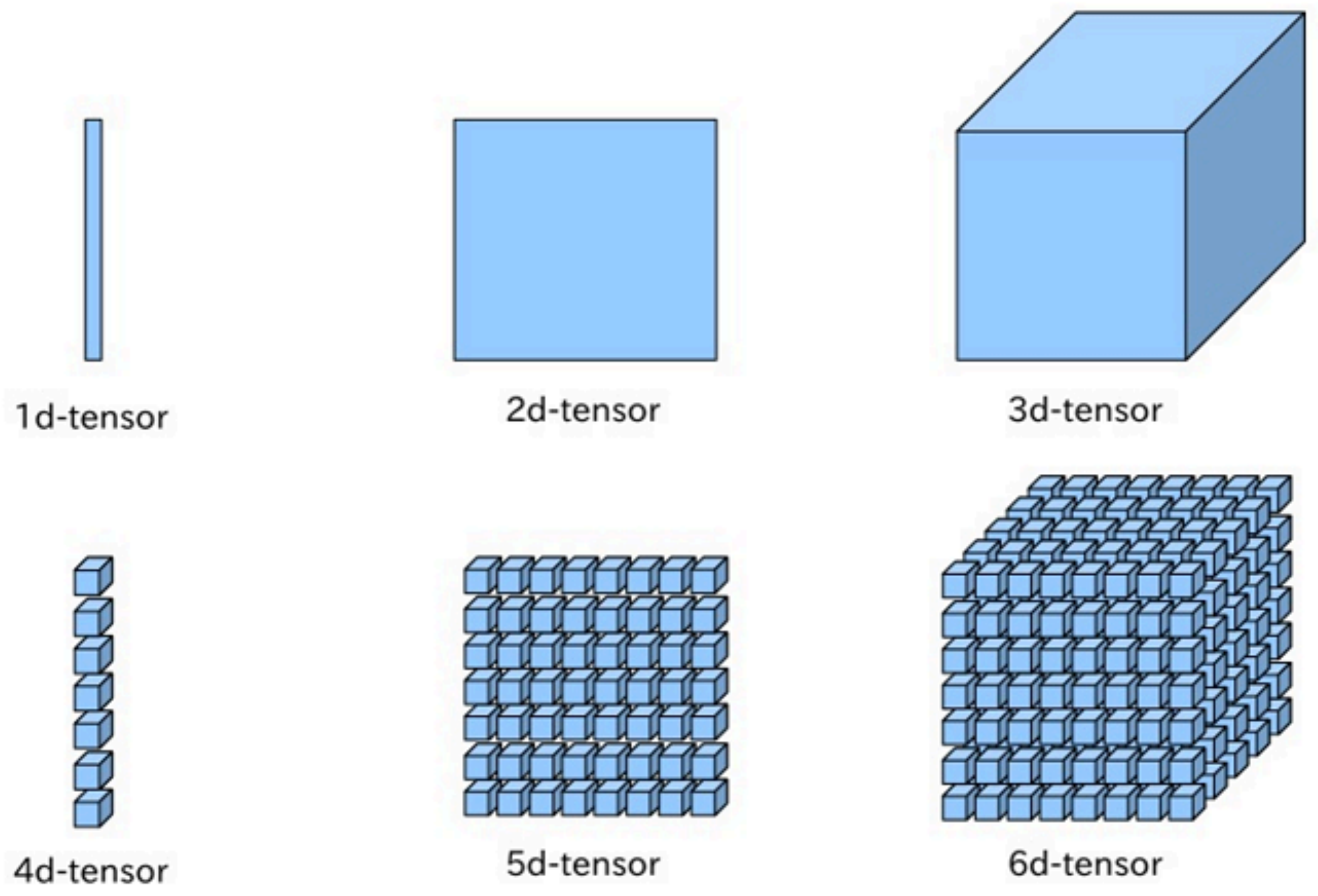
Dense & CNN

学习路线



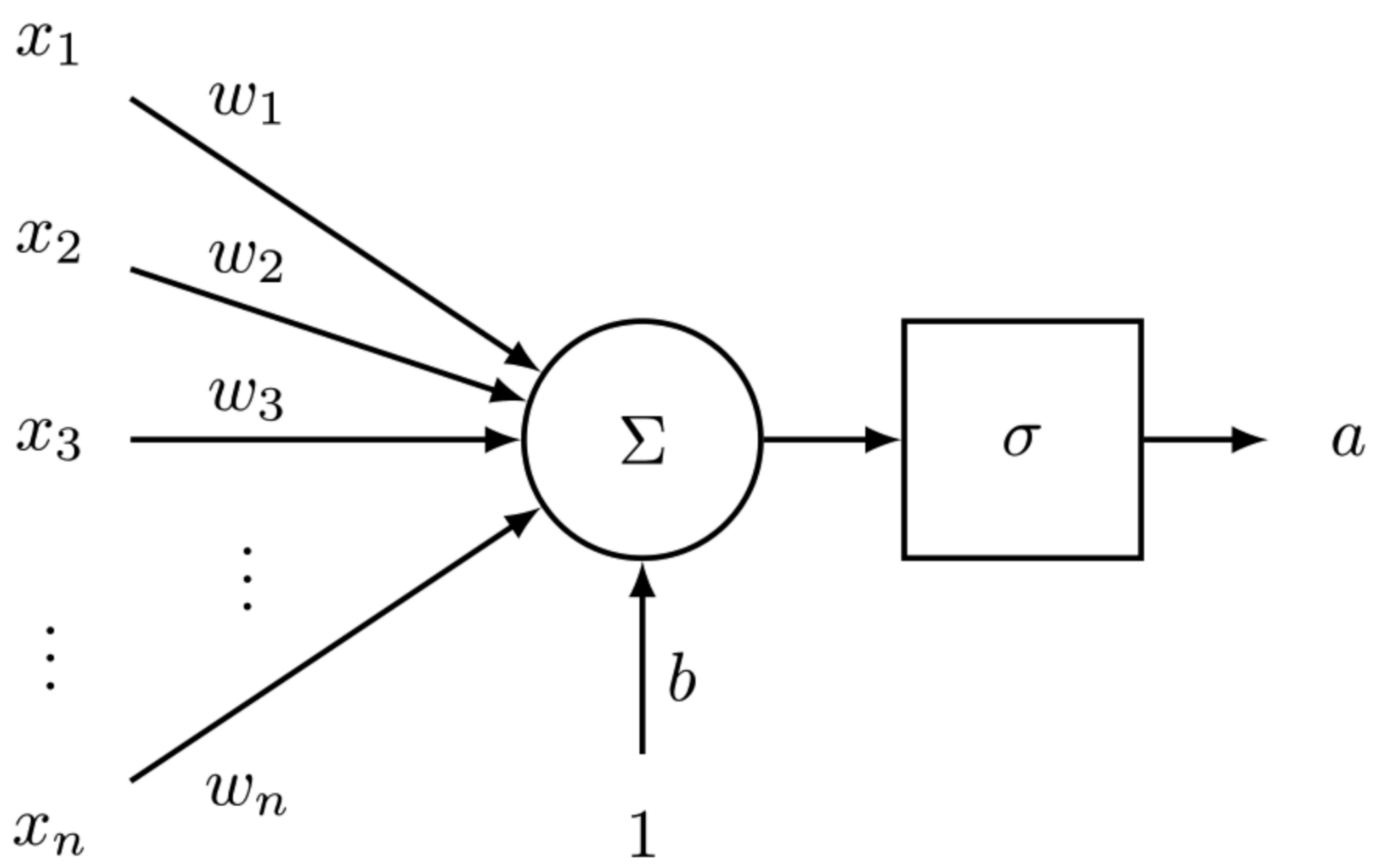
形象化Tensor

- 对于一个4*5*6的Tensor
  - rank : 3d
  - length: 4, 5, 6
  - shape: [4, 5, 6]
  - volume: 4*5*6=120



从Neuron到Layer

- 单个神经元:
  - 输入是1d, 参数是1d, 输出是0d
- 一层神经元构成一个Layer
  - 显然输出的shape和Layer的shape一致.
- batch_size
  - 会影响输出的shape
  - 并不会影响参数的shape



Dense

- 意义: 多层Dense构成MLP, 用于解分类问题.
- 排列结构: Layer的结构是1d
- 超参数: 神经元的个数U
- shape:
  - input = L
  - weights = L* U
  - output = U

```
tf.layers.dense(
    inputs,
    units,
    activation=None,
    use_bias=True,
    kernel_initializer=None,
    bias_initializer=tf.zeros_initializer(),
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    trainable=True,
    name=None,
    reuse=None
)
```

CNN组成

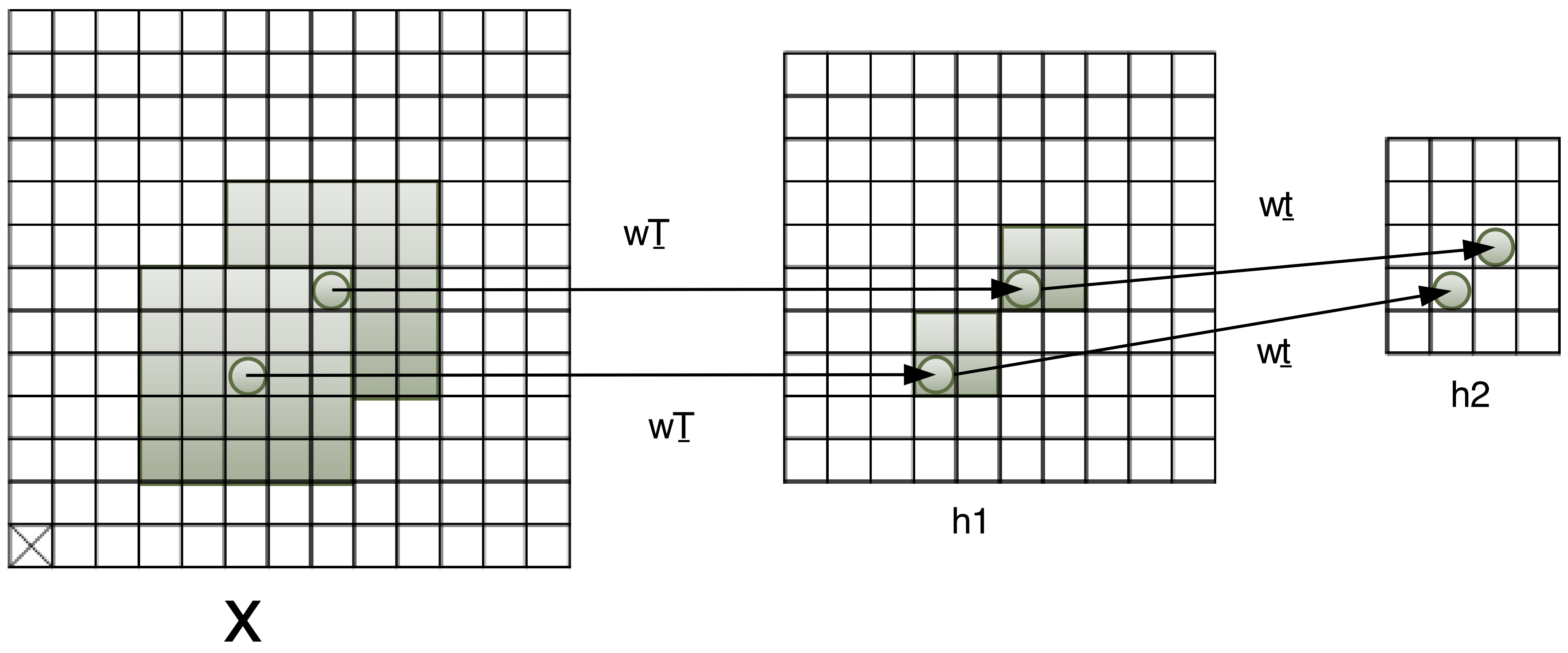
- convolutional layers 卷积层
- pooling layers 采样层
- normalization layers 正则层(dropout)
- MLP 分类器

MLP处理图像

- 考虑用MLP来处理一张图像
  - 图像单个像素点的颜色RGB值表示
  - 一张200x200x3的图片
  - 单个神经元有200*200*3 = 120,000参数!

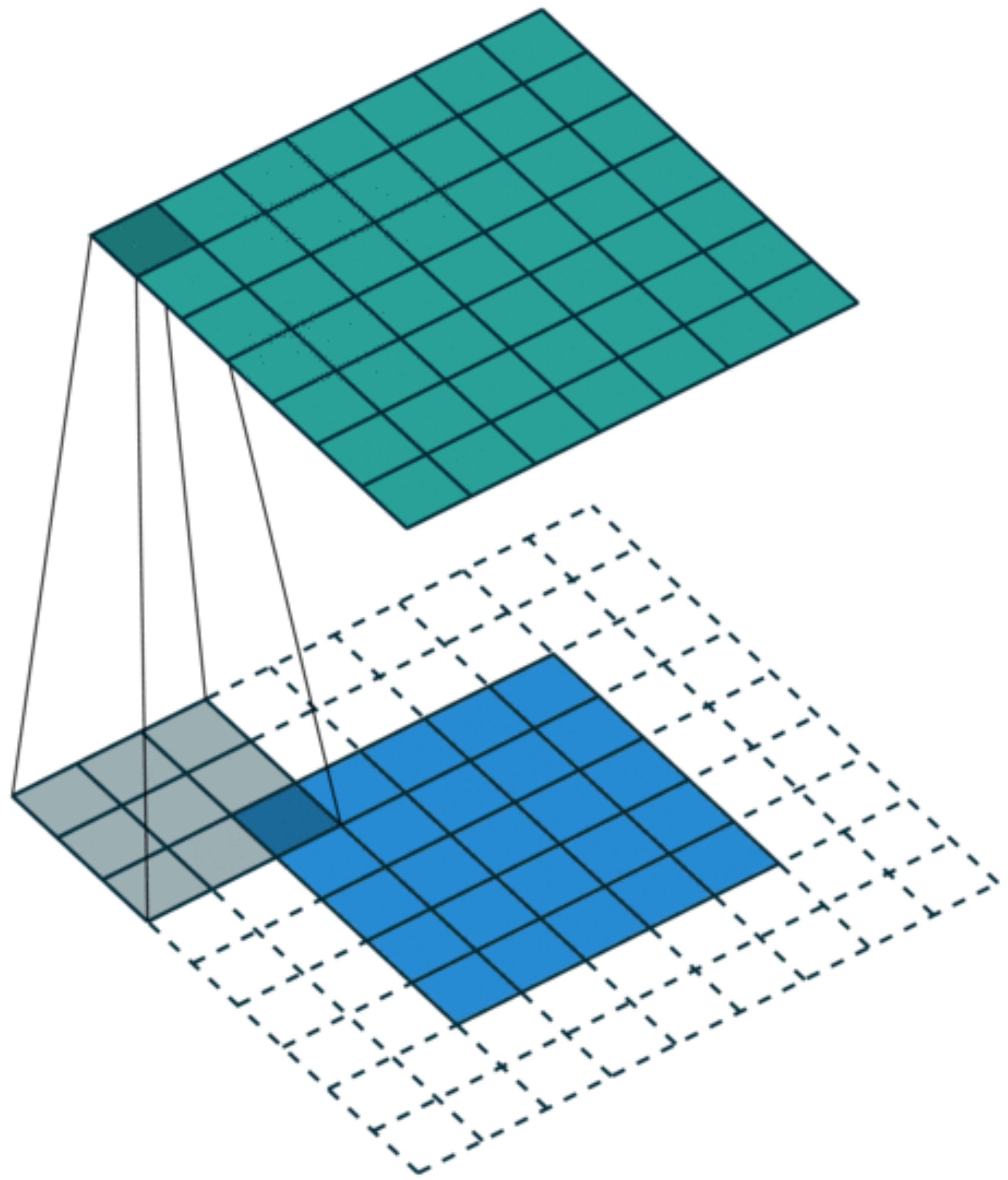
CNN

- 卷积网络（Convolutional neural network，简称CNN）
- 特点：局部区域的权重W共用（weight sharing）（空间维度）
- 每一个卷积层后通常紧跟着一个下采样层subsample，比如采用max-pooling 方法完成下采样。

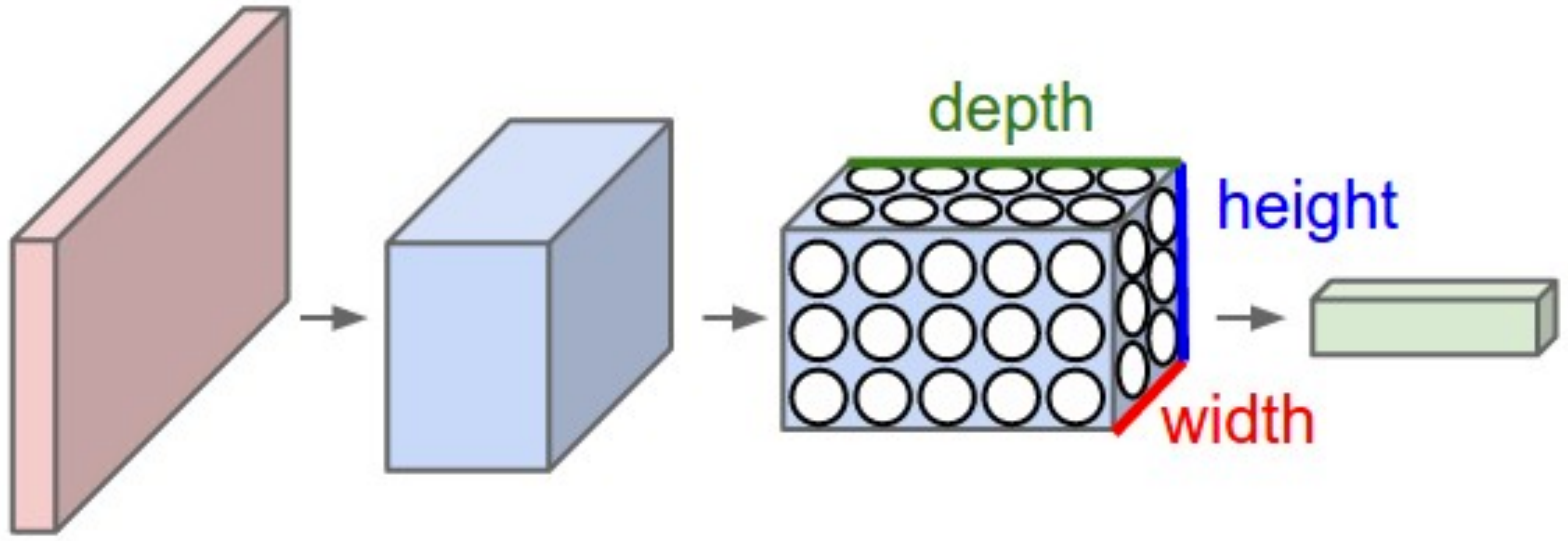


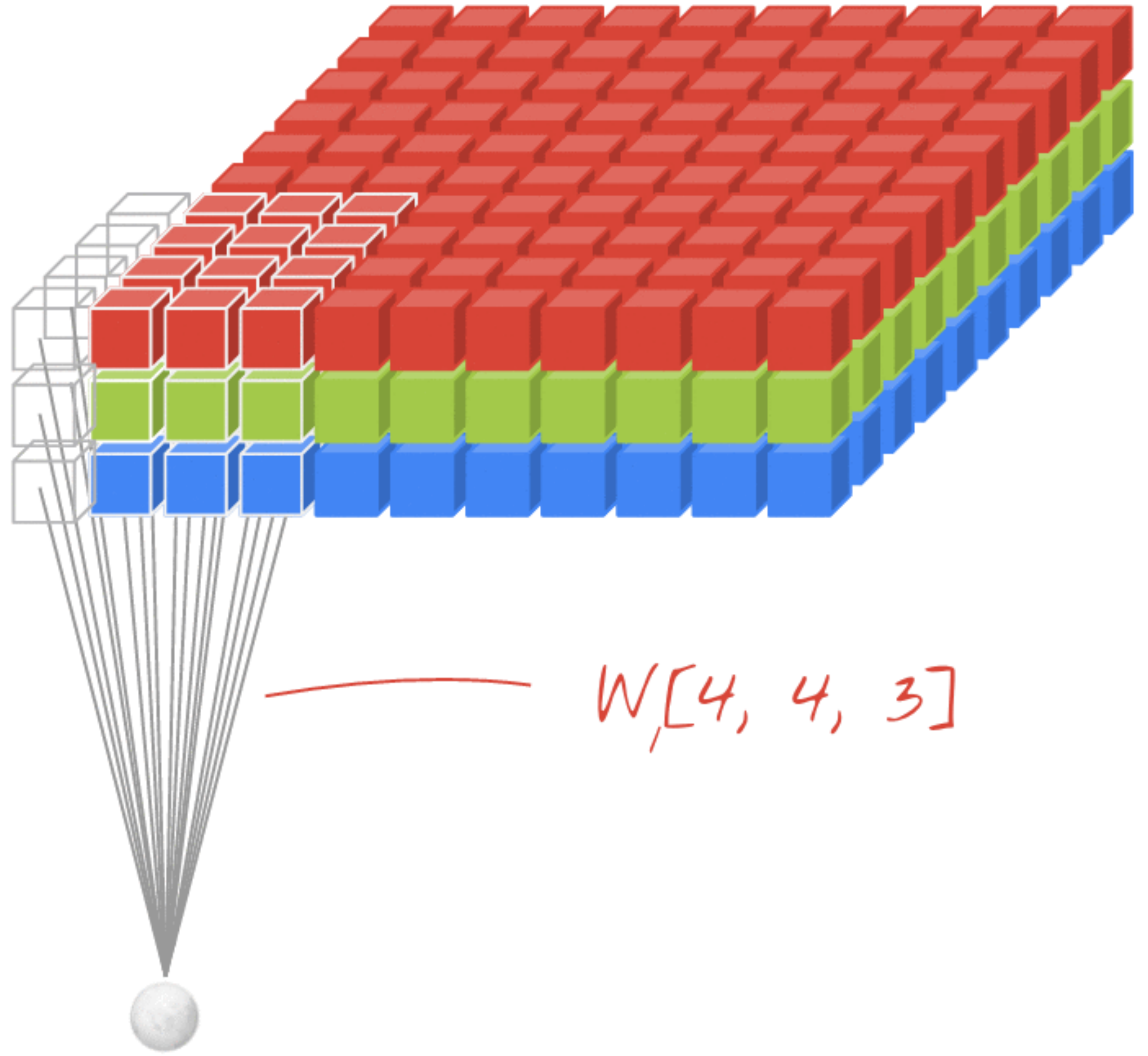
2d卷积核

- 卷积核是一个多维数组，参数由学习算法得到的
- 定义输入的长度(W), 卷积核的大小(F), 核移动的步长stride(S), zero padding(P)
- 输出的长度L = (W-F+2P)/S+1
- 并行化: 做一个和输出一样大小的Layer, Layer里面所有的神经元参数都一样!



3d卷积核





- 输入是3d的
- 有多个卷积核

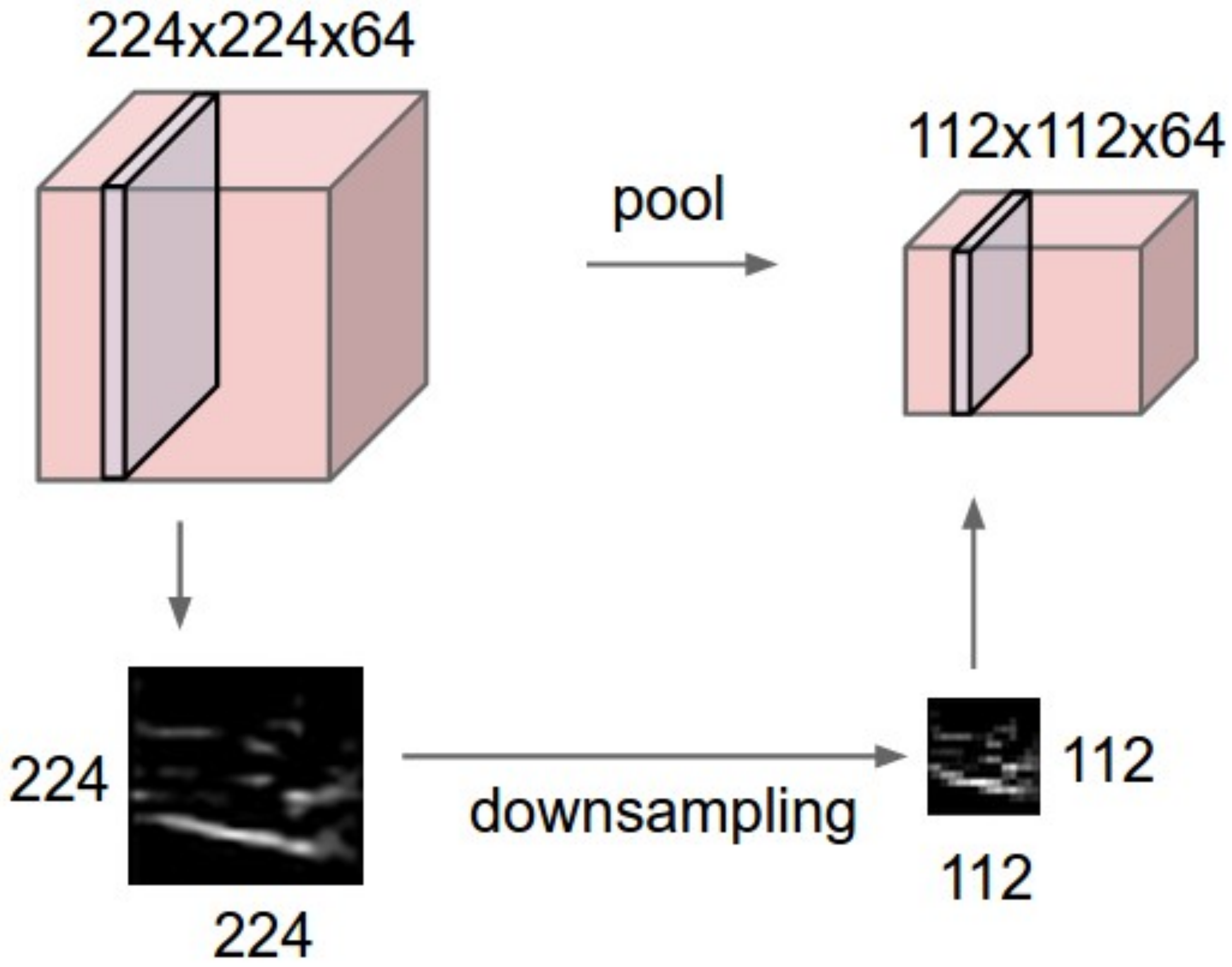
卷积层

- 意义: 用于处理图像.
- 排列结构: Layer的结构是3d
- 超参数: 卷积核个数(D), 核大小(F), padding(P), strides(S)
- shape:
    - Input = W*W*3
    - L = (W-F+2P)/S+1
    - Layer = L*L*D
    - Weights = F*F*D
    - Output = L*L*D

```
tf.layers.conv2d(
    inputs,
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format='channels_last',
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer=None,
    bias_initializer=tf.zeros_initializer(),
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    trainable=True,
    name=None,
    reuse=None
)
```

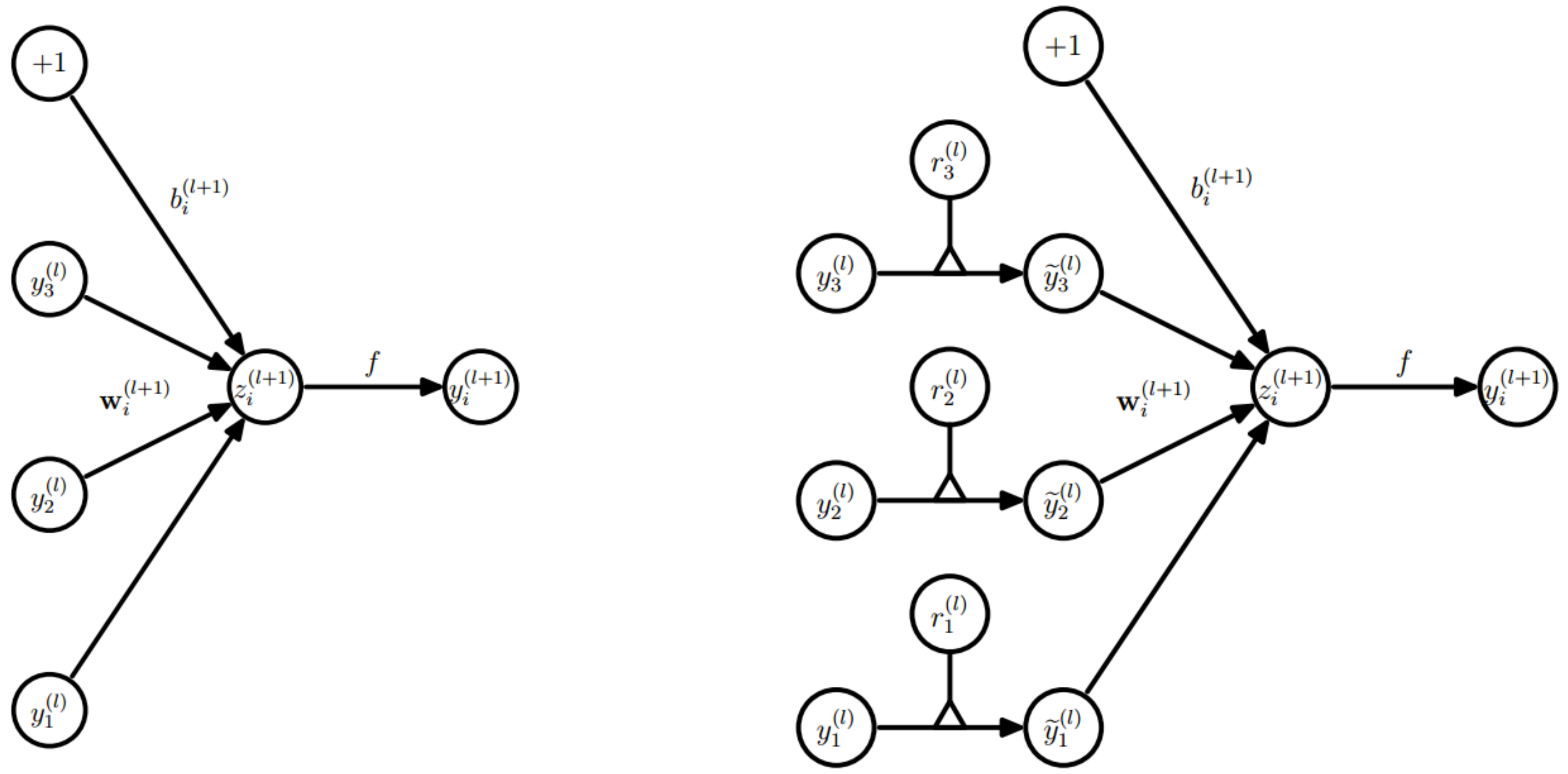
Pooling层

- 意义: 采样,缩小模型大小
- 排列结构: Layer的结构是3d
- 超参数: pooling_type, window_shape, padding, strides
- 一个2*2核, strides=2的pooling层, 等于减少75%的输出
- pooling层并不会改变tensor的深度



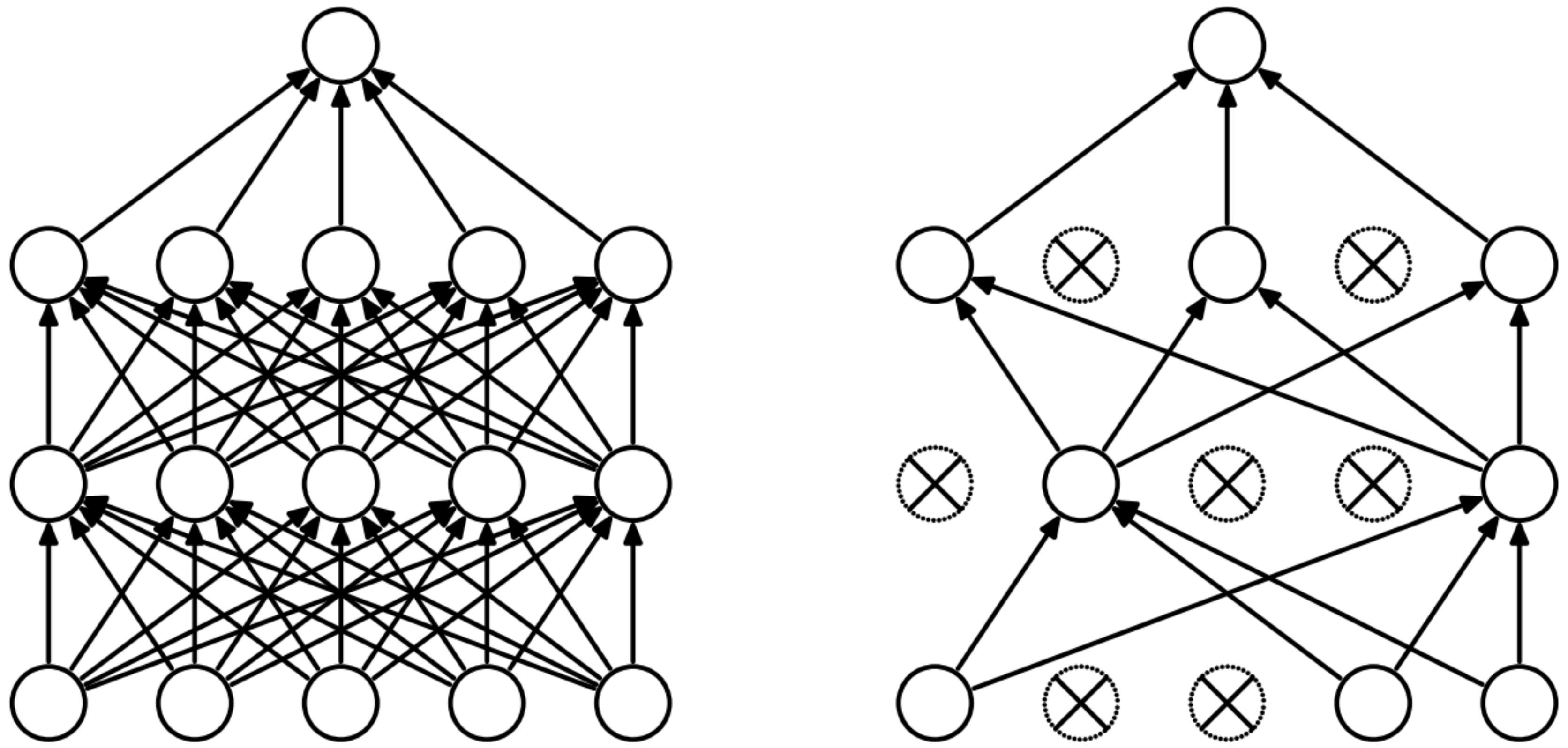
Dropout层

- 意义: 减少CNN过拟合问题
- 超参数: keep_prob 丢弃率
- 对于所有的输入, 有keep_prob概率保留并乘以1/keep_prob, 以保证前后总和大致相等, 否则输出0



(a) Standard network

(b) Dropout network



(a) Standard Neural Net

(b) After applying dropout.

共享Variable

```python
def my_image_filter(input_images):
    with tf.variable_scope("conv1"):
        # Variables created here will be named "conv1/weights", "conv1/biases".
        relu1 = conv_relu(input_images, [5, 5, 32, 32], [32])
    with tf.variable_scope("conv2"):
        # Variables created here will be named "conv2/weights", "conv2/biases".
        return conv_relu(relu1, [5, 5, 32, 32], [32])
```

```python
with tf.variable_scope("model") as scope:
  output1 = my_image_filter(input1)
  scope.reuse_variables()
  output2 = my_image_filter(input2)
```

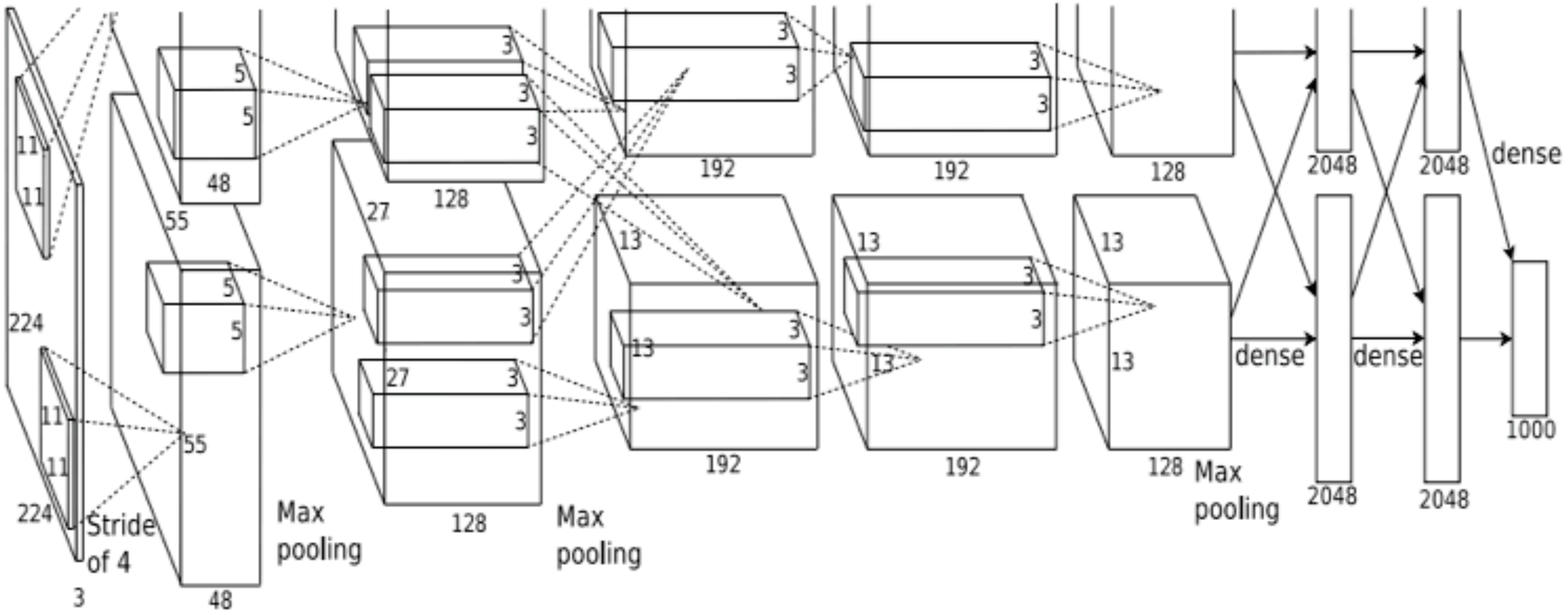
生成两套参数

```python
with tf.variable_scope("model") as scope:
  output1 = my_image_filter(input1)
  scope.reuse_variables()
  output2 = my_image_filter(input2)
```

共享一套参数

https://www.tensorflow.org/programmers_guide/variables#sharing_variables

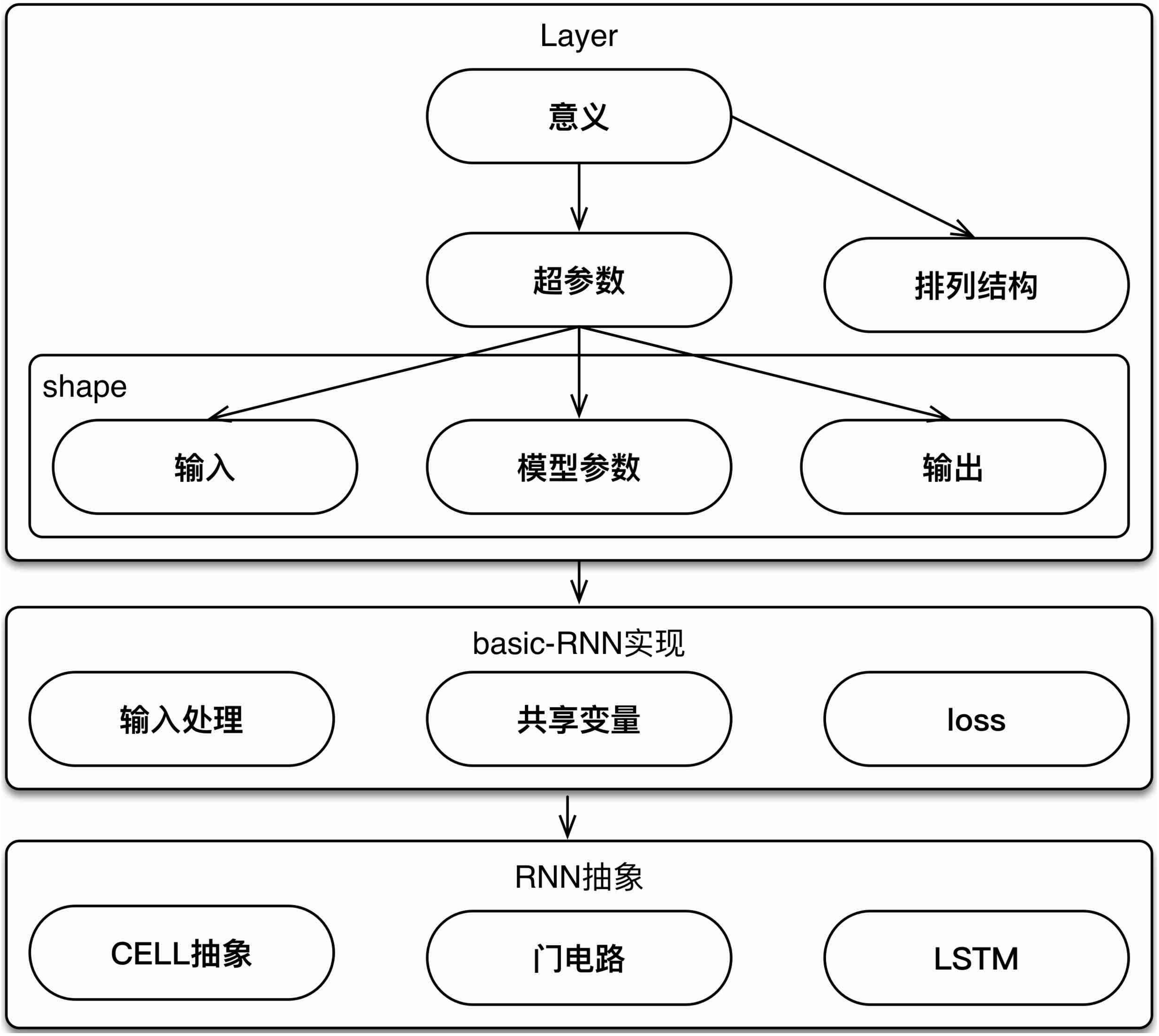
课后作业



阅读 Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." NIPS 2012.

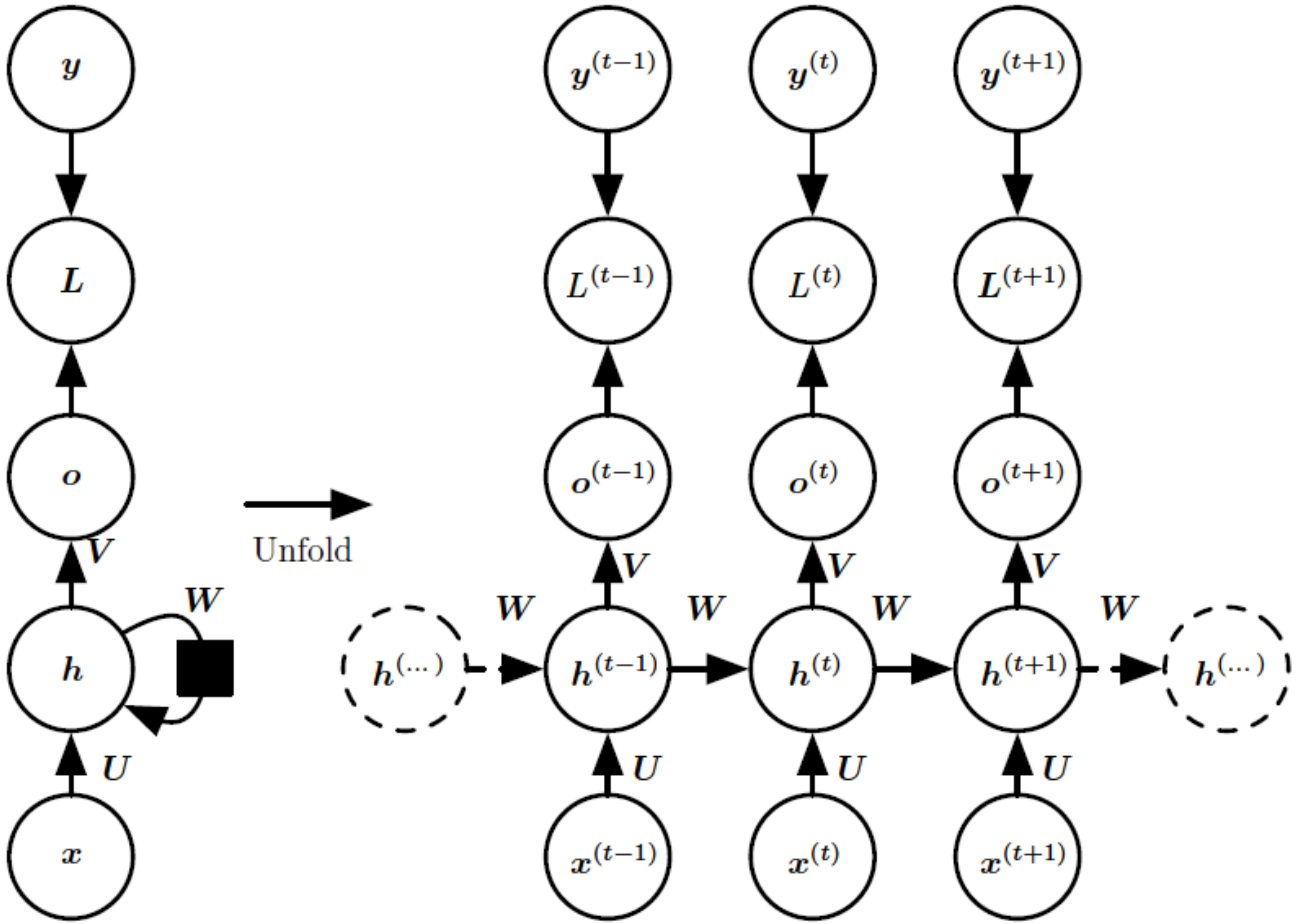
RNN & LSTM

学习路线


Layer
意义
超参数
排列结构
shape
输入
模型参数
输出
basic-RNN实现
输入处理
共享变量
loss
RNN抽象
CELL抽象
门电路
LSTM

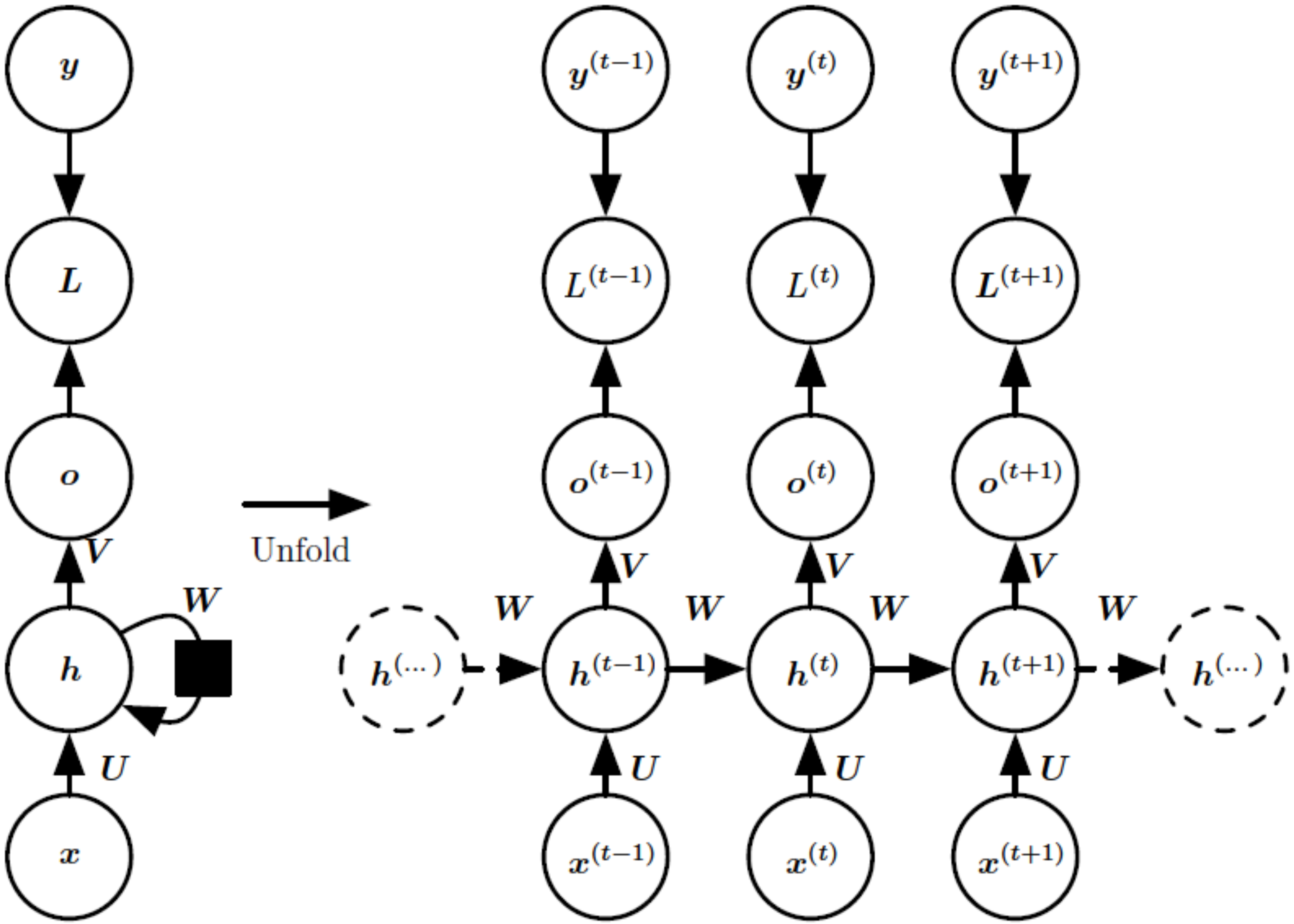
RNN

- 循环网络结构
  - y是训练目标
  - L是损失函数
  - o是网络输出
  - h是状态（隐藏单元）
  - x是网络输入
- 计算图的时间步上展开
- 举例：天气预测



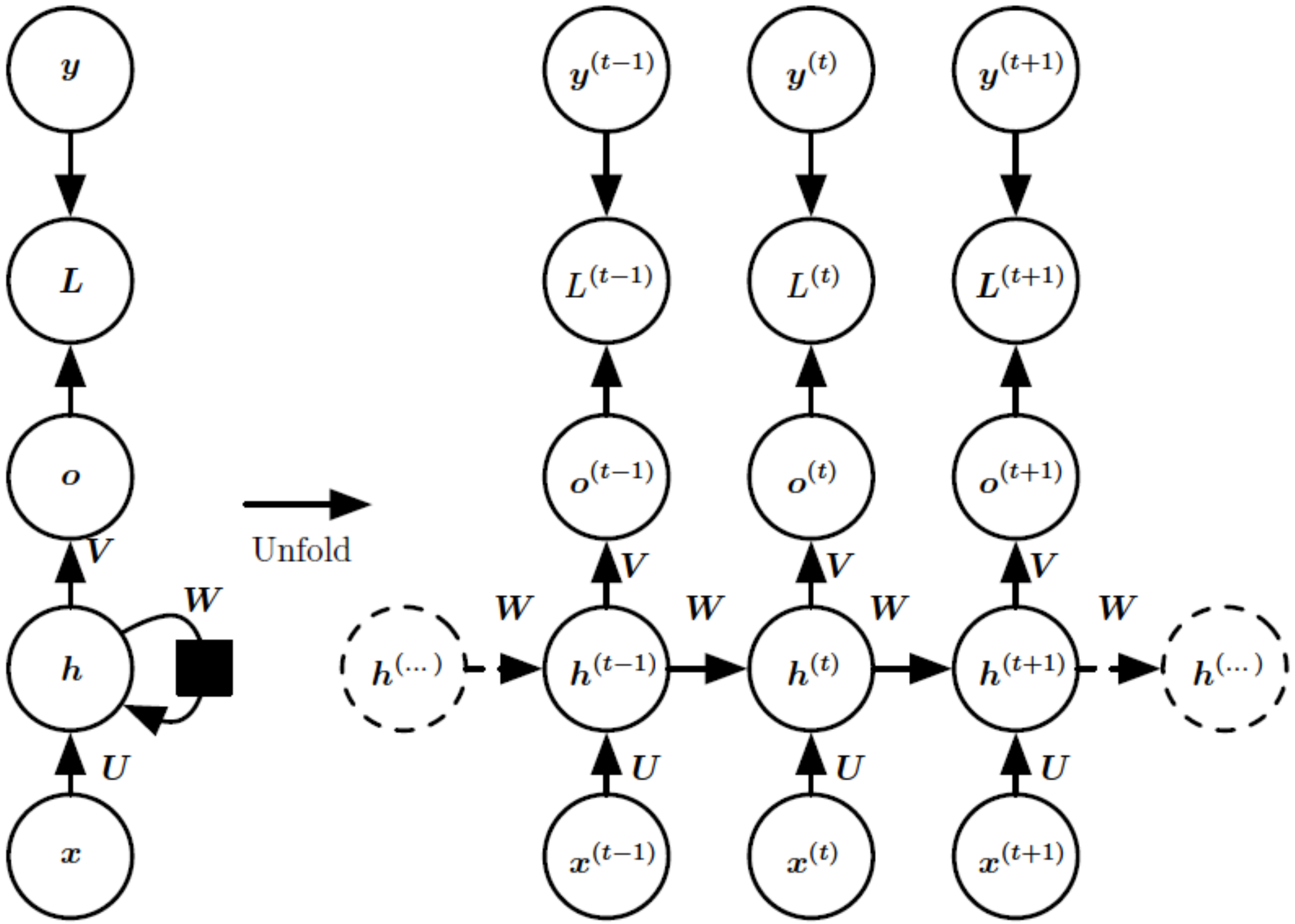
权重共享

- 循环神经网络在不同的时间步上采用相同的U、V、W参数
- 输入到隐藏的连接由权重矩阵U 参数化
- 隐藏到输出的连接由权重矩阵V 参数化
- 隐藏到隐藏的循环连接由权重矩阵W 参数化



计算图

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)},$$
$$h^{(t)} = \tanh(a^{(t)}),$$
$$o^{(t)} = c + Vh^{(t)},$$
$$\hat{y}^{(t)} = \mathrm{softmax}(o^{(t)}),$$



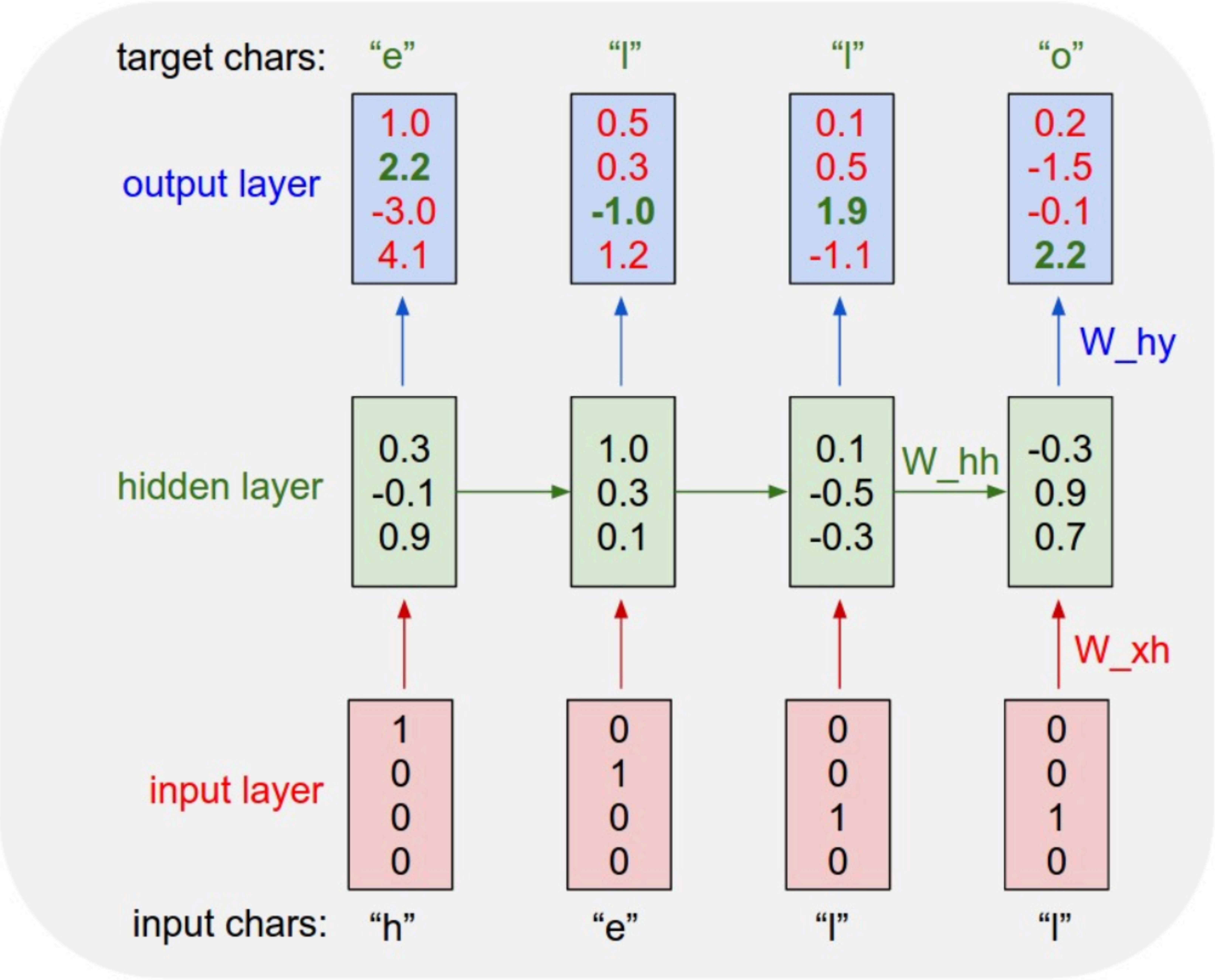
- 循环网络将一个输入序列映射到相同长度的输出序列。
- 信息流动路径：信息在时间上向前（计算输出和损失）和向后（计算梯度）的思想。
- U、V 和W分别对应于输入到隐藏、隐藏到输出和隐藏到隐藏的连接的权重矩阵。
- b 和c 是偏置向量。

basic-rnn 实现

```
iter: 646500, p: 1300, loss: 0.019042
----
 y = tf.get_variable("by", [vocab_size], initializer=initializer)
        by   = tf.get_variable("by", [vocab_size], dtype=tf.float32, name="state:
 hprev_val = np.zeros([1, hidden_size])

while True:
```

- <u>3_NeuralNetworks/min-char-rnn-tensorflow.py</u>
- Andrej Karpathy的min-char-rnn tf版本实现
- 实现了一个自动写代码的程序, 输入程序就是本身



target chars:
“e”
“l”
“l”
“o”
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
“h”
“e”
“l”
“l”

输入和loss处理

- 给定序列长度(模型超参数), 把输入序列化
- one-hot 离散化处理
- U, V, W 共享权重
- 收集所有时刻的输出,计算的loss
- 梯度截取预防梯度爆炸

$$
\begin{aligned}
& L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right) \\
& = \sum_t L^{(t)} \\
& = -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right)
\end{aligned}
$$

rnn-cell抽象

```
__init__(
    num_units,
    activation=None,
    reuse=None,
    name=None
)
```

Cell超参数: num_units

```
__call__(
    inputs,
    state,
    scope=None,
    *args,
    **kwargs
)
```

调用时刻要输入state

```
tf.nn.static_rnn(
    cell,
    inputs,
    initial_state=None,
    dtype=None,
    sequence_length=None,
    scope=None
)
```

static-rnn抽象

```
state = cell.zero_state(...)
outputs = []
for input_ in inputs:
  output, state = cell(input_, state)
  outputs.append(output)
return (outputs, state)
```

rnn-example

https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/BasicRNNCell

rnn-cell抽象

- hidden-units : 模型的容量大小
- I(input) + S(state) -> O(output) + S(new_state)
- inputs: 输入
- state: 隐含了之前所有的输出信息
- 当前的输出完全取决于state和当前的输入

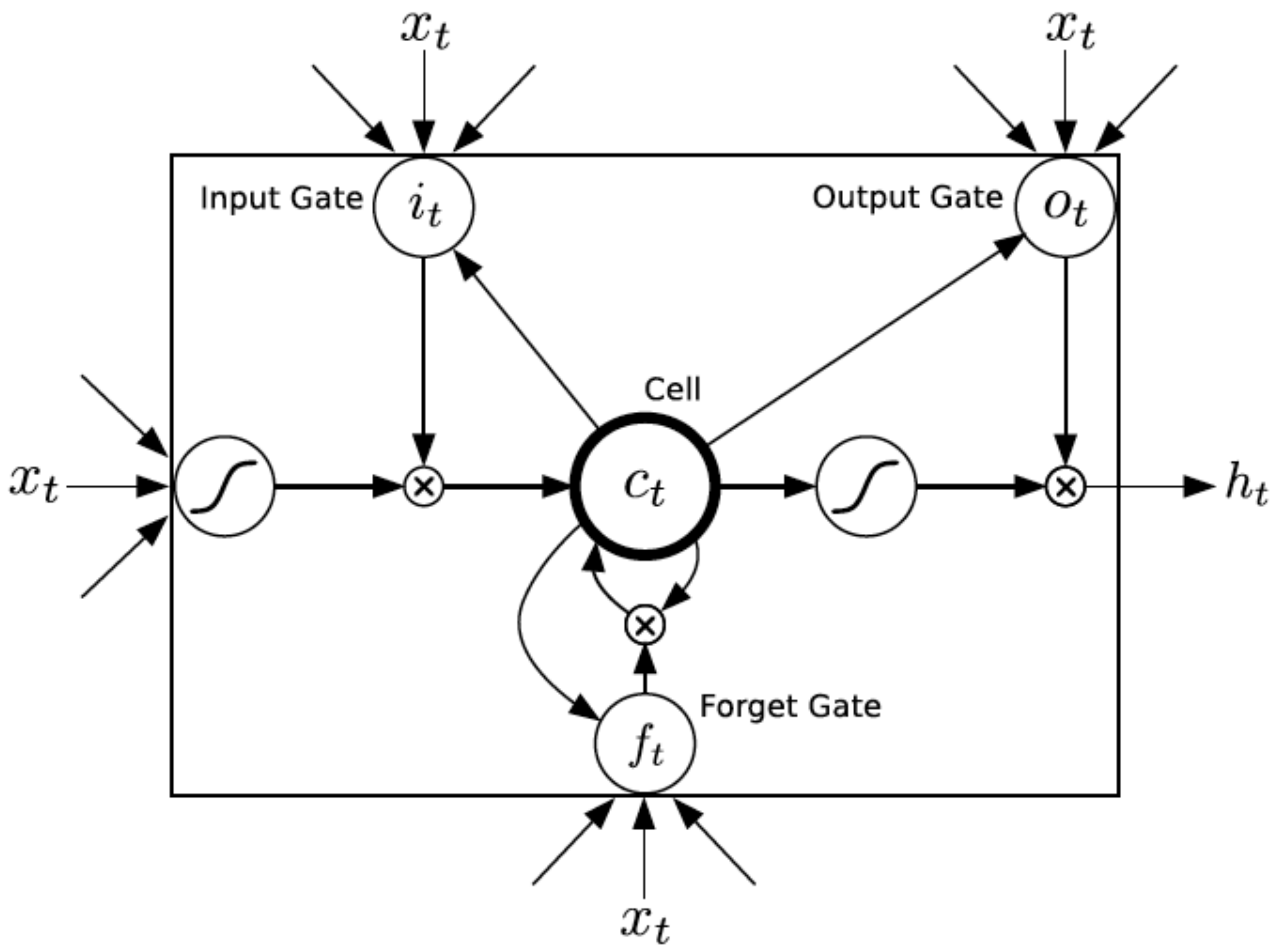
dynamic-rnn

- 3_NeuralNetworks/dynamic_rnn.py
- 动态生成计算图
- 不需要固定序列长度

```
tf.nn.dynamic_rnn(
    cell,
    inputs,
    sequence_length=None,
    initial_state=None,
    dtype=None,
    parallel_iterations=None,
    swap_memory=False,
    time_major=False,
    scope=None
)
```

LSTM

- RNN训练有以下问题
  - RNN梯度爆炸
  - RNN梯度消失
- LSTM解决以上问题

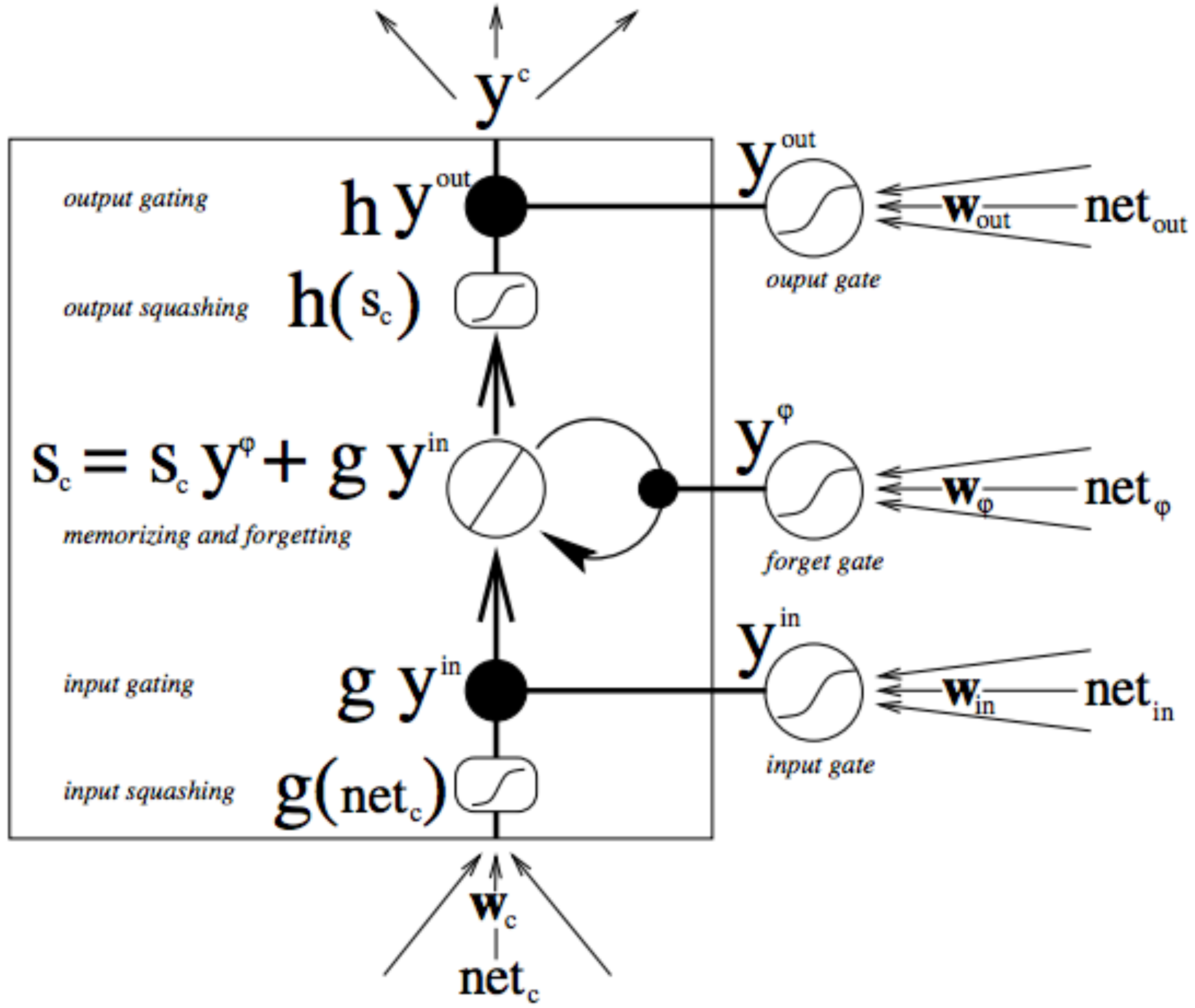


LSTM

- LSTM是RNN的一个改进，LSTM增加了一个主输入单元和其他三个辅助的门限输入单元：输入门（Input gate）控制是否输入，遗忘门（Forget gate）控制是否存储，输出门（Output gate）控制是否输出。

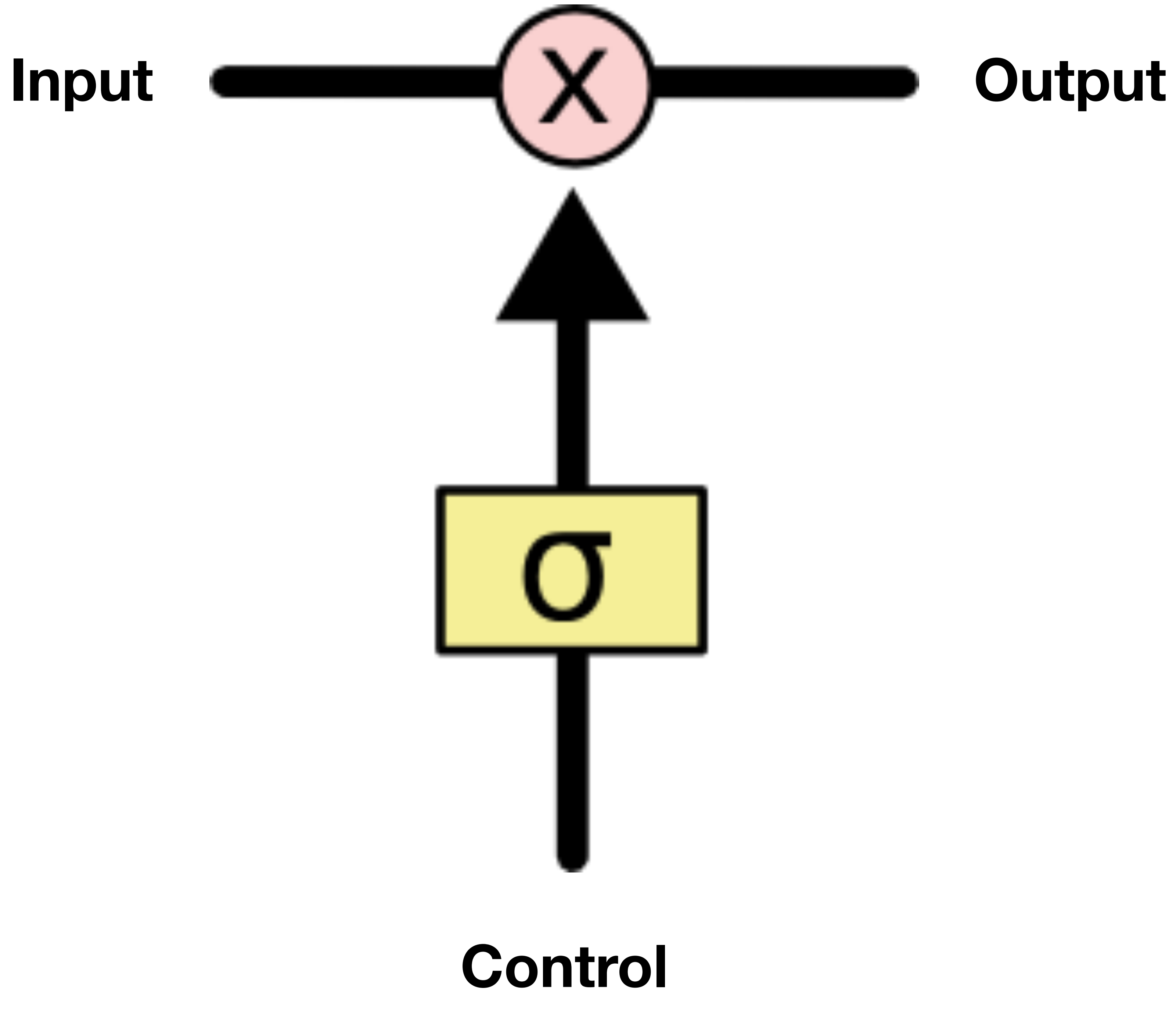
- 辅助单元可以寄存时间序列的输入，在训练过程中会利用后向传播的方式进行。

- 记忆单元和这些门单元的组合，大大提升了RNN处理远距离依赖问题的能力，解决RNN网络收敛慢的问题。



门电路

- Input和Control形状一致
- Control经过Sigmoid函数后, 变成一个范围在0-1之间的一个同形状的Tensor
- Input和σ(Control) 元素相乘等到一个同形的Output



- RNN

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right) \qquad (1)$$

$$y_t = W_{hy}h_t + b_y \qquad (2)$$

- LSTM

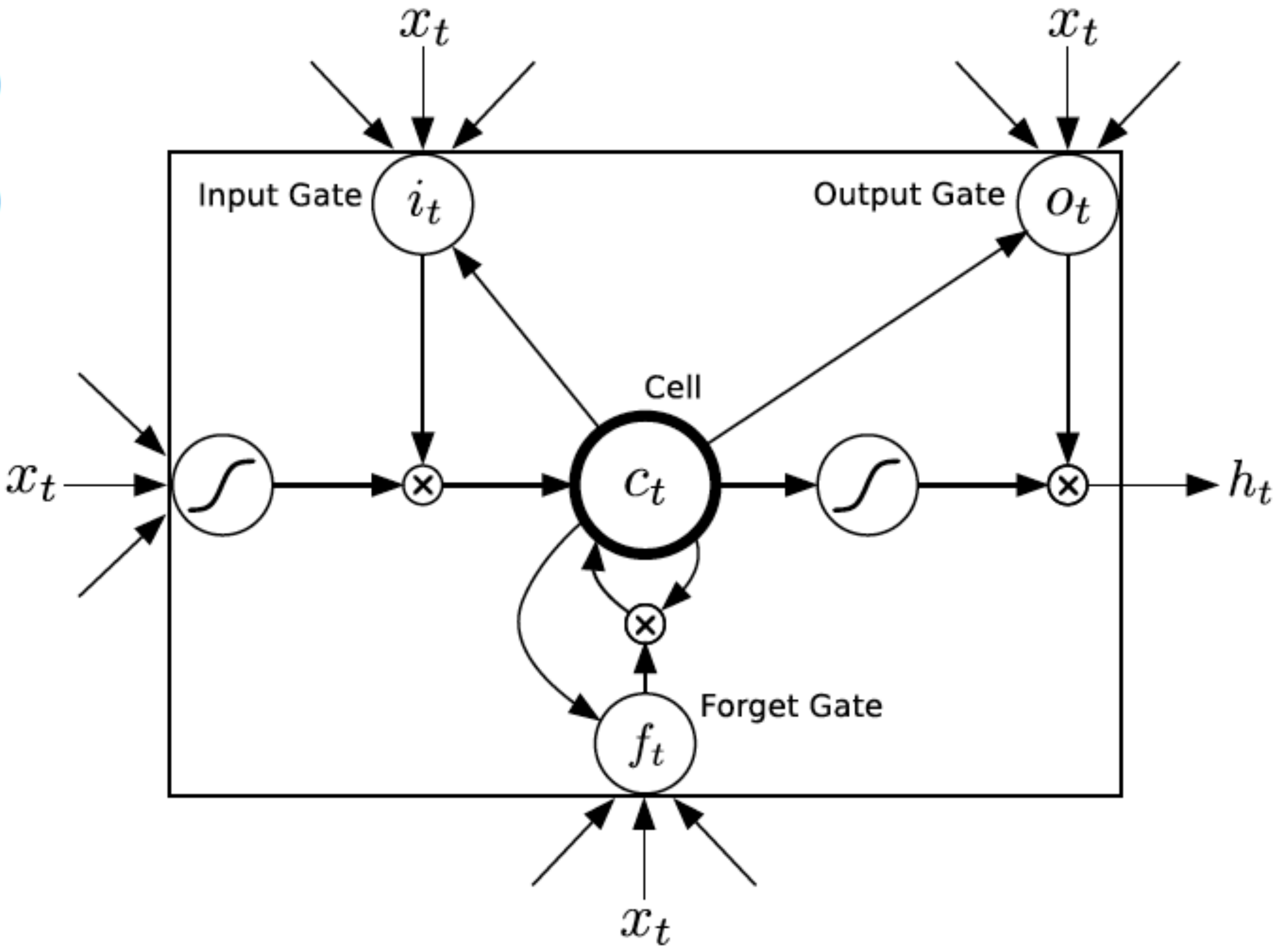
$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \qquad (3)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \qquad (4)$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \qquad (5)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \qquad (6)$$

$$h_t = o_t \tanh(c_t) \qquad (7)$$



推荐阅读

- The Unreasonable Effectiveness of Recurrent Neural Networks
- Understanding-LSTMs

谢谢!