卷积网络/循环网络

Convolution Networks/Recurrent Networks

陈震

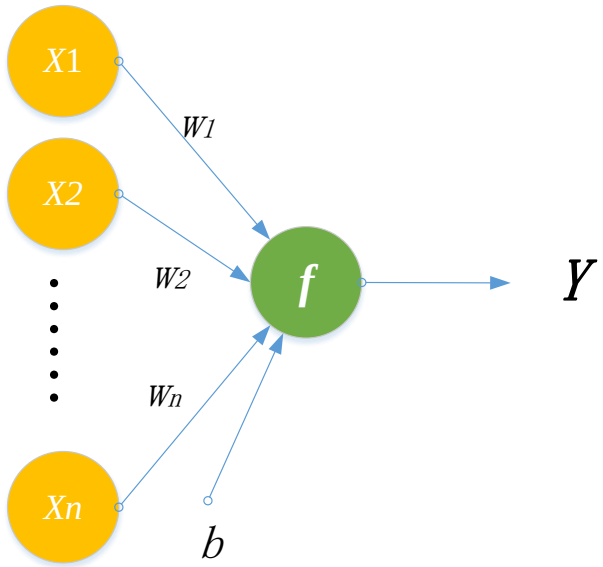
清华大学iCenter

人工神经网络

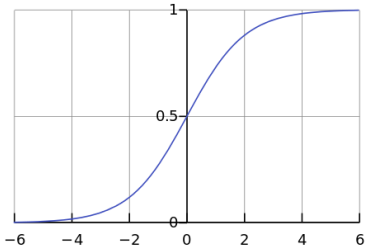
Artificial Neural Networks

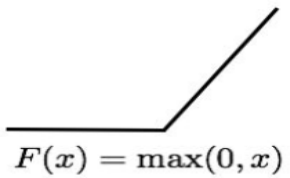
人工神经元(Artificial Neuron)

- 单个神经元可以理解为一组输入线性加权叠加，再经过一个非线性变换进行输出。

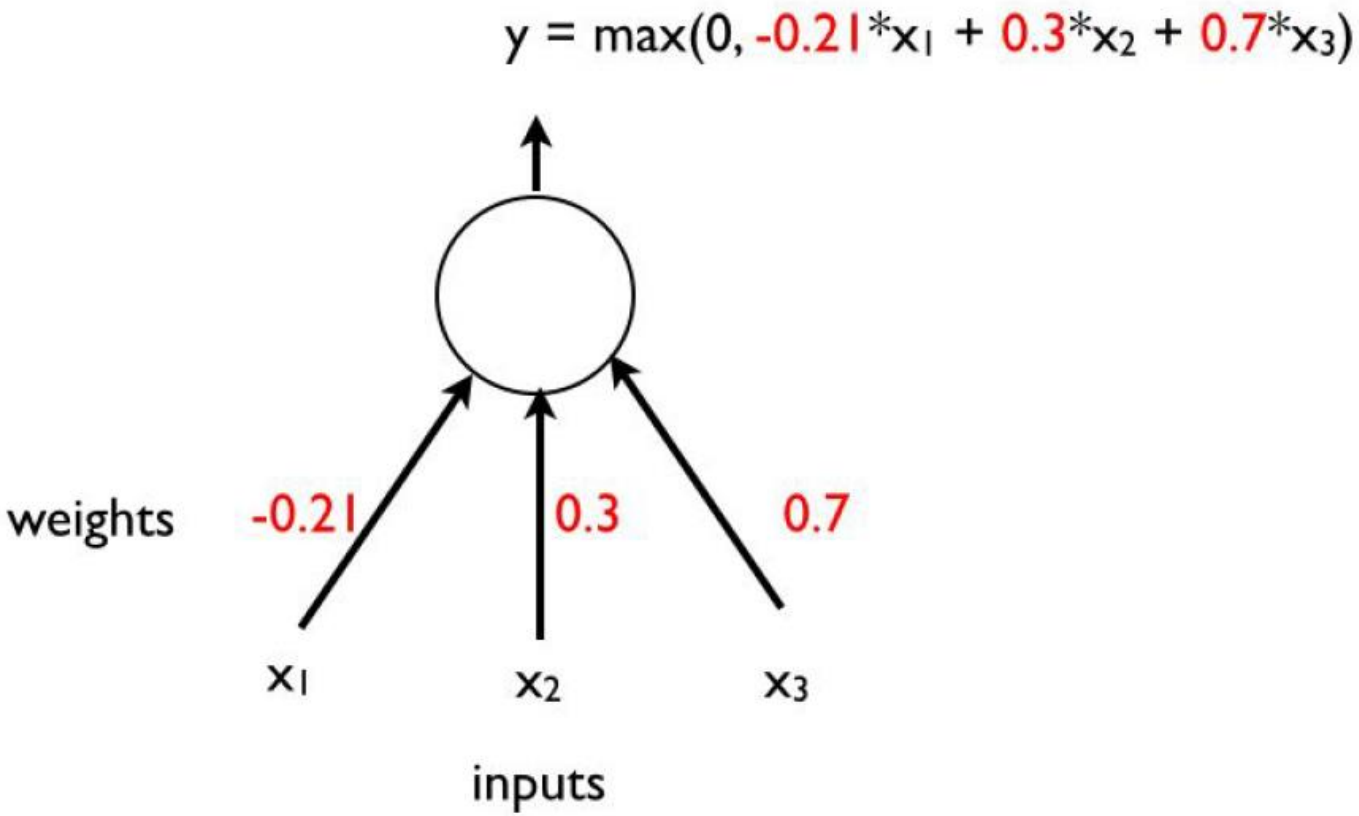


$$y = f\left(\sum_{i=1}^{N} W_i X_i + b\right)$$



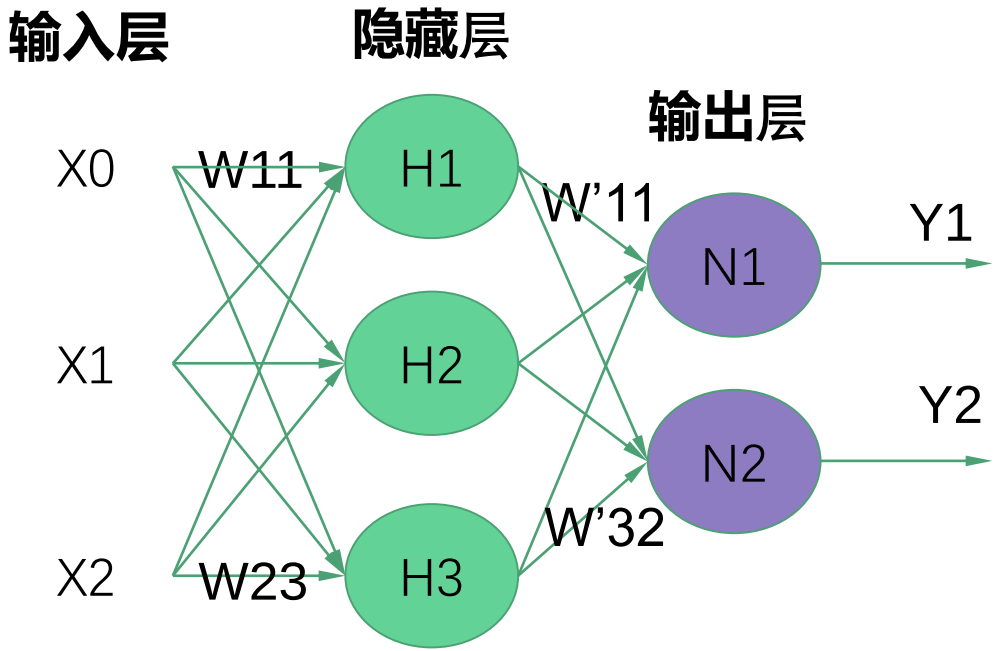


调节权重(weights)



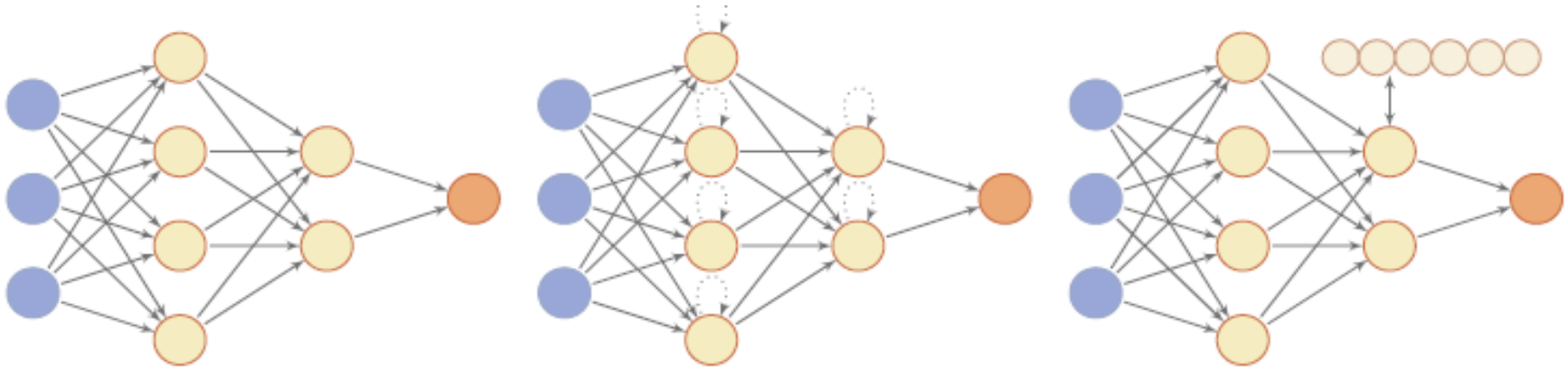
多层人工神经网络（Multilayer neural networks）

- 单个神经元可以进行线性分类，而多个神经元的组合就可以完成复杂的分类工作。
- 多层神经网络，又称为深度神经网络，在实际中表现出更好的性能。



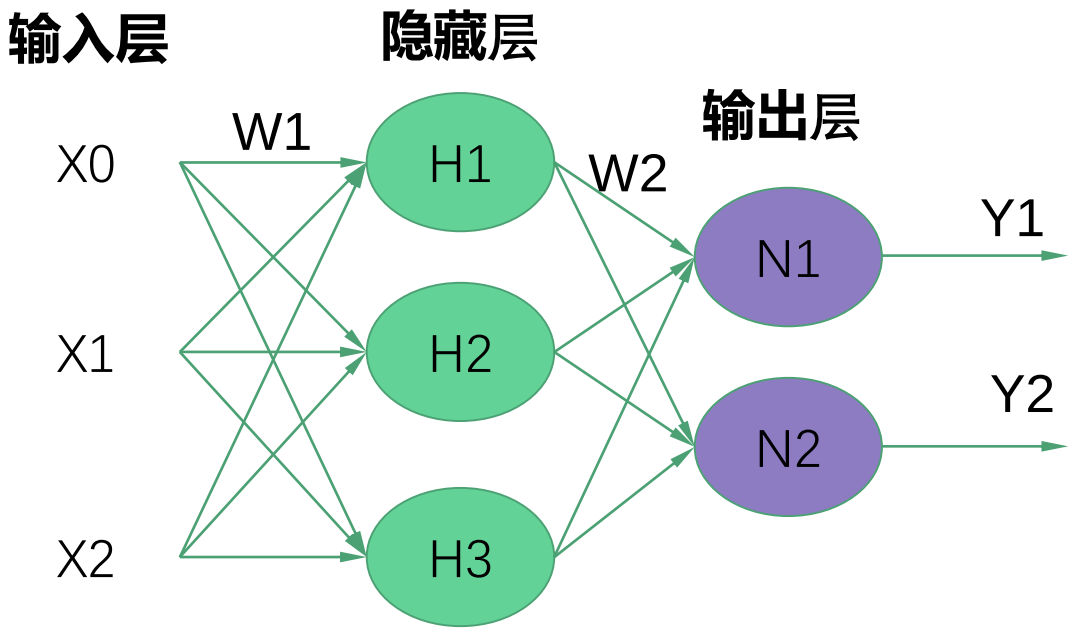
多层人工神经网络的层间连接关系

- 按照连接主义观点：人工神经网络由大量的神经元以及它们之间的有向连接构成，能够实现复杂的智能功能。
- 网络的拓扑结构：不同层神经元之间的连接关系。
  - 前馈网络（feedforward），反馈网络（feedback）和记忆网络（memory network)



多层前馈网络（Multilayer feedforward networks）

- 前馈网络结构（feedforward）是一种计算图（Computational Graph）
- 别名：多层全连接网络（FCN）、多层感知机（MLP）、密集网络（Dense）
- 神经网络有3个输入，2个输出，中间有1个隐藏层，有1个输出层，共有5个神经元。
- 对应存在反馈的网络，如循环网络等，反馈网络依然展开为计算图



logit

- 分对数（logit）

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

$$\forall x \in (0, 1), \sigma^{-1}(x) = \log\left(\frac{x}{1 - x}\right)$$

Softmax处理

- 输出层的Softmax 处理，计算出一个概率分布:

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

- 所有分量之和为 1,所有输出的数值是正的。

参见python-tensorflow-basics

人工神经网络工作原理

- 人工神经网络方法是机器学习中的监督学习（supervise learning）
- 监督学习过程分为训练和推断两阶段，其中训练数据集的样本（sample）上都有相应的标签（label）。
- 神经网络训练的本质是找到相应的内部权重参数，使得在训练数据（样本）输入到网络后，网络的实际输出与预期输出（即标签）之间差异的度量函数取得最小值。
- 用训练得到网络参数（固化的网络参数），对训练数据集外的数据预测其可能的标签。

神经网络训练-模型

- 采用带标签的训练样本对神经网络进行学习，确定神经网络内部的**权重参数**
  - 数据集：(x1,y1), (x2,y2), (x3,y3), …,
- 神经网络输出
  - 实际输出：(x1, y1’), (x2, y2’), (x3, y3’), …,
  - 其中：Y’为实际输出
- 实际输出与预期输出（标签）的差异
  - Y为预期输出（标签）
  - 预测集与标签：(y1, y1’), (y2, y2’), (y3, y3’), …,
- 度量函数的差异
  - 度量函数：绝对值求和，平方和，交叉熵等
  - 目标优化（最小化）：min{(|y1-y1’|+ |y2-y2’|+ |y3-y3’|+…)}

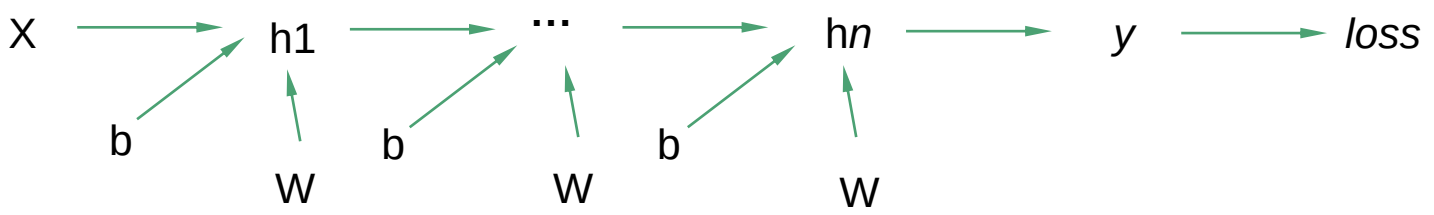
神经网络训练-损失函数（Loss）

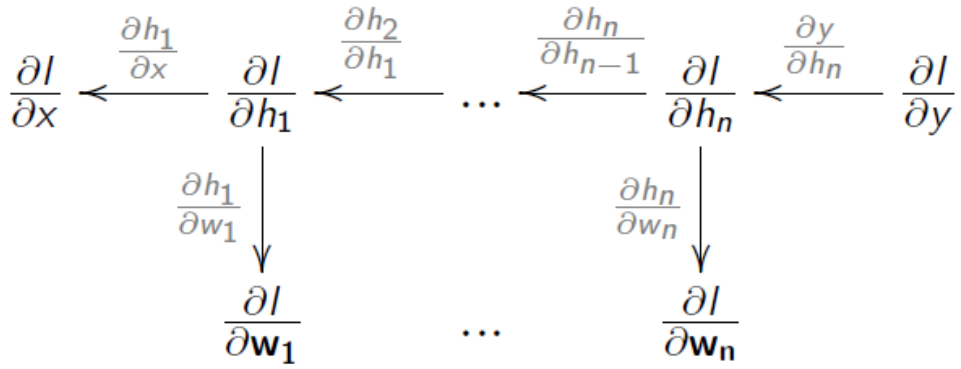
- 神经网络的输出结果会与预期结果（标签）之间的存在差别。
- 引入度量函数来数值化这种差异，叫做损失函数(Loss)或成本函数(Cost)。
- 损失函数的具体方式：
  - 对于回归任务，通过均方误差的公式，来计算损失。
  - 对于分类任务，通过交叉熵的公式，来计算损失(Loss)。
  - 交叉熵损失函数表示如下：其中y'表示训练样本对应的标签，y表示神经网络的输出。

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

神经网络训练-反向传播(BP)

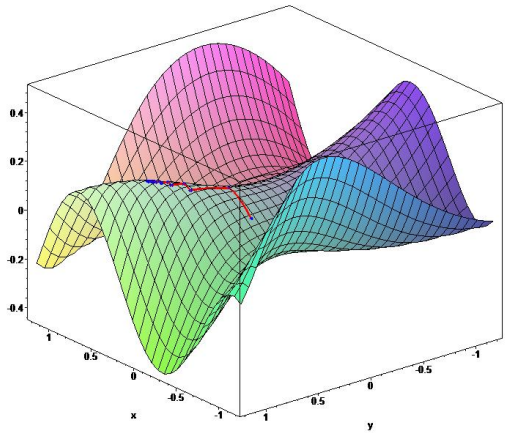
- 根据损失函数的性质以及链式求导法则，反向逐层计算损失函数对**权重**的梯度（各个偏导数），这个过程称为反向传播算法。
- 多层神经网络的一种抽象表示，这种表示抽象了每层基本特征。

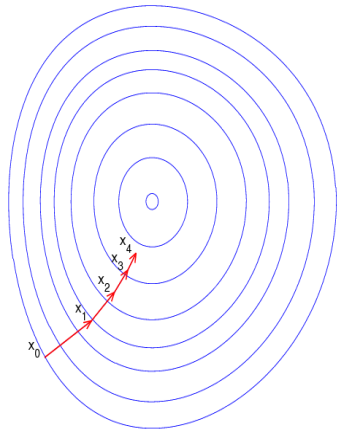




梯度下降法（Gradient descent）

- 梯度下降法也称为最速下降法，是一个最优化算法。
- 梯度的定义：
  - 函数F(x)在某个点上增长变化率最大的方向，就是导数的方向。
- 梯度下降法的原理：
  - 如果实值函数F(x)在点 a 处可微且有定义，那么函数F(x) 在a 点沿着梯度相反的方向下降最快。
- 梯度下降法的过程：
  - 找到一个函数的局部极小值，必须向函数上当前点对应梯度的反方向，按照规定步长距离点，进行迭代搜索。





梯度下降法-参数值更新（梯度算子）

- 在原值θ的基础上，沿着梯度方向的步长，这样就得到了一个新的θ数值，使得损失函数变化的最快。
- 当不断调整θ数值，Loss(θ)是一个随着θ取值的序列，这个序列的数值不再有大的变化，就说明收敛了。

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta).$$

神经网络的实际训练过程

- 神经网络的训练将样本数据“分批训练”。
- 最简单的批量选择是使用整个数据集，又称为批量训练（batch training）。
- 优点：
  - 批量训练的收敛性是有保证的。
  - 共轭梯度法和L-BFGS加速技术在批量训练中效果显著
- 缺点：
  - 模型参数的更新前，需要遍历整个数据集

随机梯度下降法(stochastic gradient descent, SGD)

- 随机梯度下降方法：
  - 是最常用的权重调节方法，通过权重的调整，最小化损失函数。
- 随机梯度下降法步骤如下：
  - 步骤 1. 随机初始化每个神经元输入权重和偏差（weights and bias）；
  - 步骤 2. 选取一个随机样本(samples)；
  - 步骤 3. 根据网络的输出结果，从最后一层开始后，逐层计算每层权重的偏导数；
  - 步骤 4. 逐层调整每层的权重，产生新的权重值。
  - 返回到步骤2，继续随机选取下一个样本。
- 随机梯度下降法的核心是一个“随机样本”

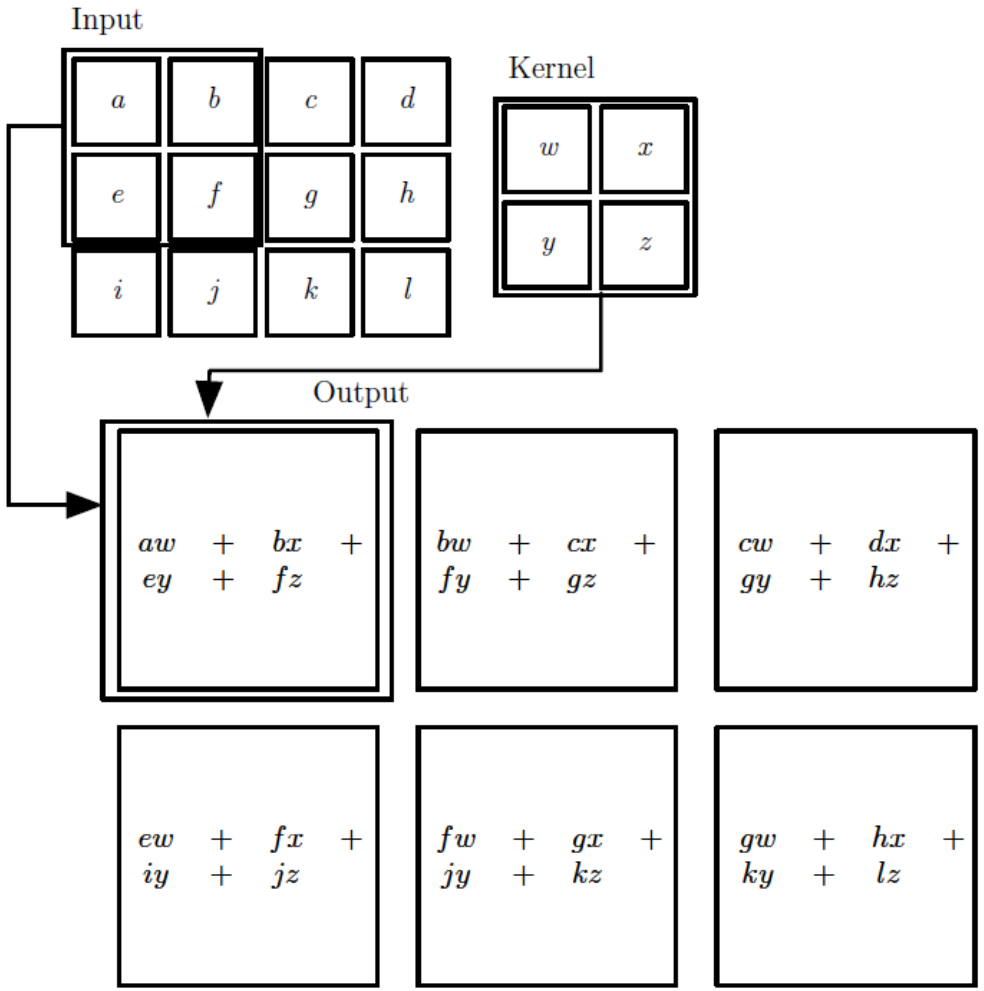
小批量训练或迷你批量训练

- 小批量训练是批量训练和随机梯度下降法的一个折中
  - 整个训练集称为一个批次（batch），先将整个训练集分成多个大小相同的子集，每个子集称为一个迷你批次（mini-batch）。子集的大小由参数迷你批次大小（mini-batch_size）控制。
  - 每个批次的数据被依次送入网络进行训练。训练完一个迷你批次， 被称为一次迭代（iteration）。
  - 一个时代（epoch）是指训练集中所有训练样本都被送入网络，完成一次训练的过程。
- 其中每一步的模型参数的更新使用
  - 使用不止一个样本，这称为迷你批次（mini-batch）。
  - 每次迭代所用的样本数目称为迷你批次大小（mini-batch size）。
  - 当迷你批次大小为1时，就退化为普通的随机梯度下降法。

卷积网络

Convolution Networks

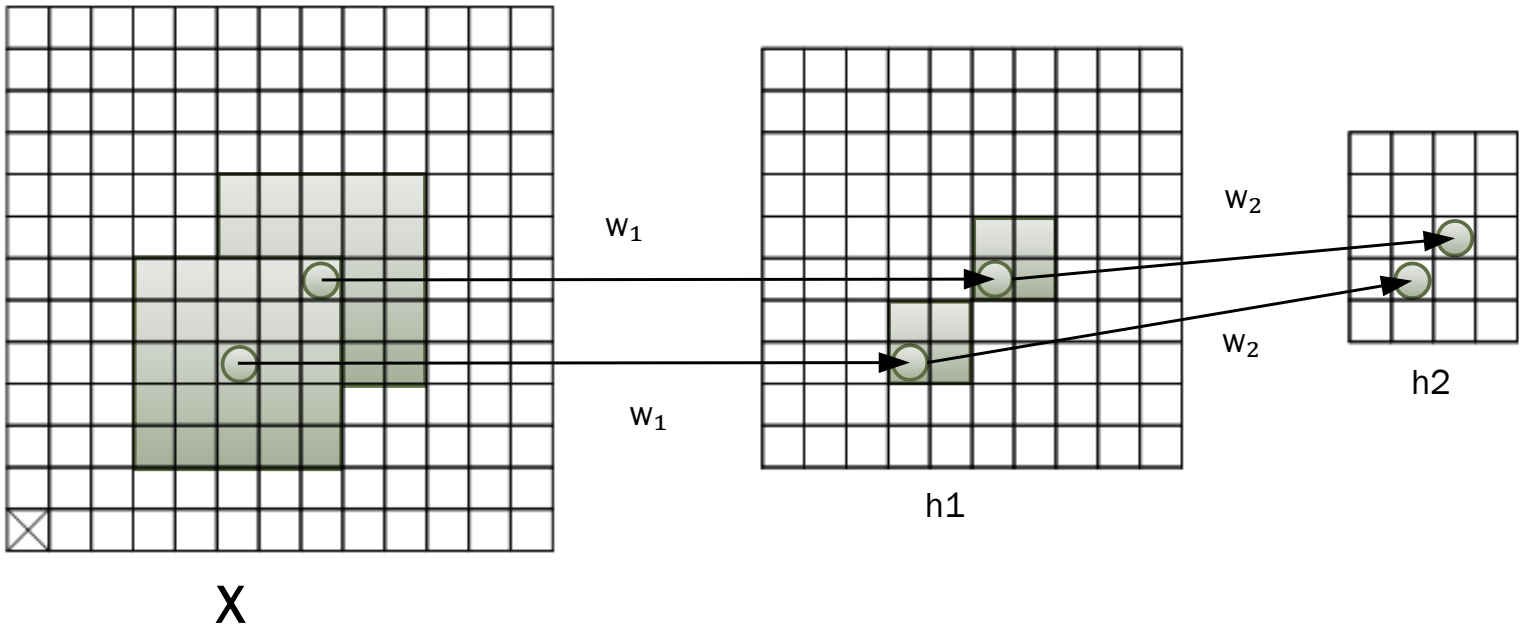
卷积运算（Convolution）



- 输入是多维数组的数据
- 卷积核是一个多维数组，参数由学习算法得到的
- 卷积核数目一般选32、64等
- 这些多维数组都是张量（Tensor）。

卷积网络

- 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。



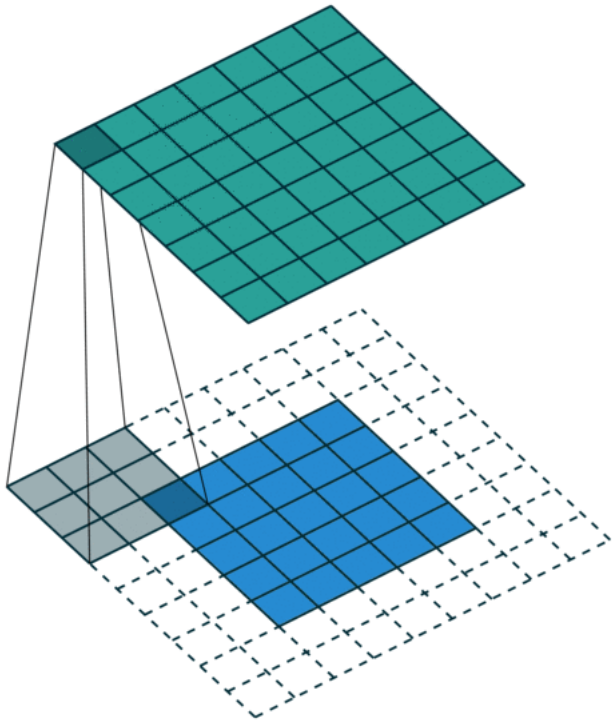
卷积网络

- 权重共享（Parameters sharing）和池化（pooling）操作
  - 利用了波形或图像信息的局部和谐性（ local coherence），学习其中的不变量特征（ invariant features）

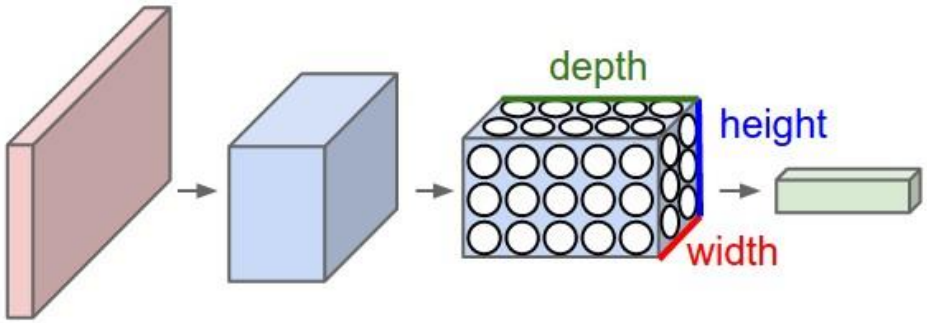
- 卷积网络的基本的结构（a series of stages of the form）：
  - Convolution/bias/non-linearity activation（ReLU or sigmoid functions)/pooling
  - 有时需要添加进去正则化层（Normalization layers），如LCN(local contrast normalization).
  - LCN操作在最大池化层之后，其目标是减去平均值，除以标准差。LCN操作具有亮度不变性的特点，对于图像识别用处很大。

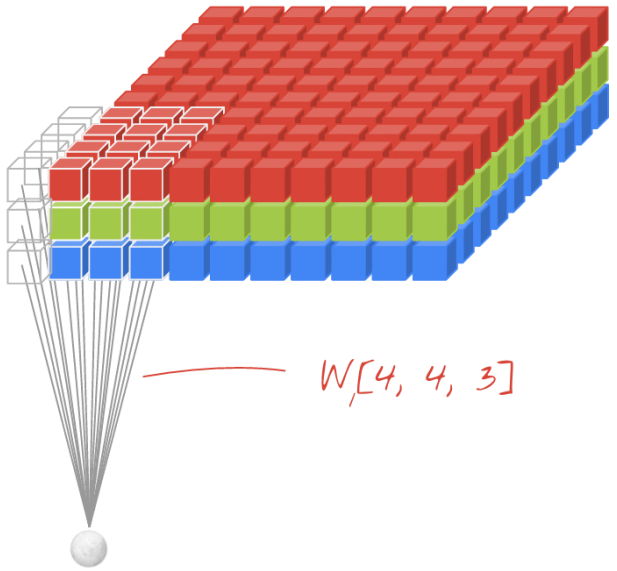
2d卷积核

- 卷积核是一个多维数组，其中参数由学习算法得到的
- 输出长度计算:
  - 输入的长度(W),
  - 卷积核的大小(F),
  - 卷积核移动的步长stride(S),
  - 填充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!



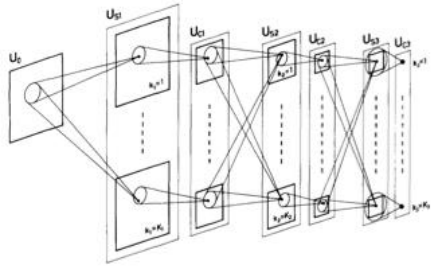
3d卷积核





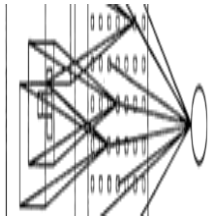
- 输入是3d的
- 有多个卷积核

卷积网络发展



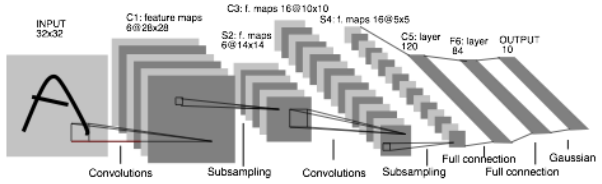
Fukushima 1980

Neocognitron



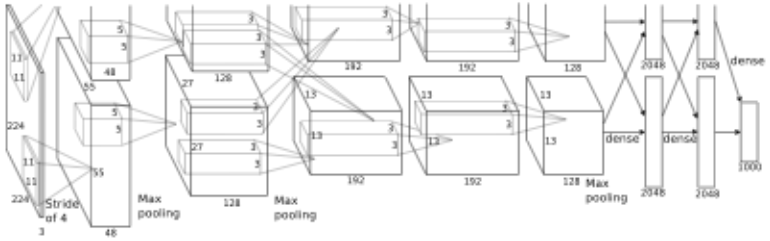
Rumelhart, Hinton, Williams 1986

“T” versus “C” problem



LeCun et al. 1989-1998

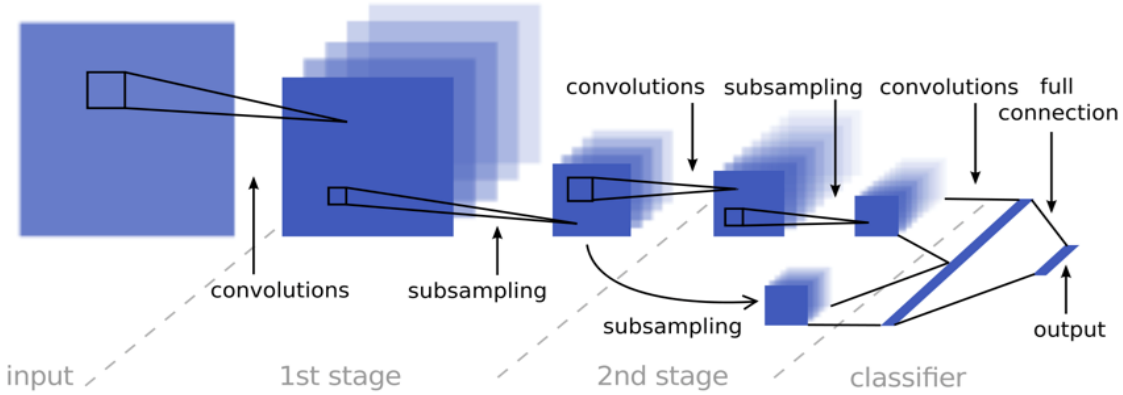
Hand-written digit reading

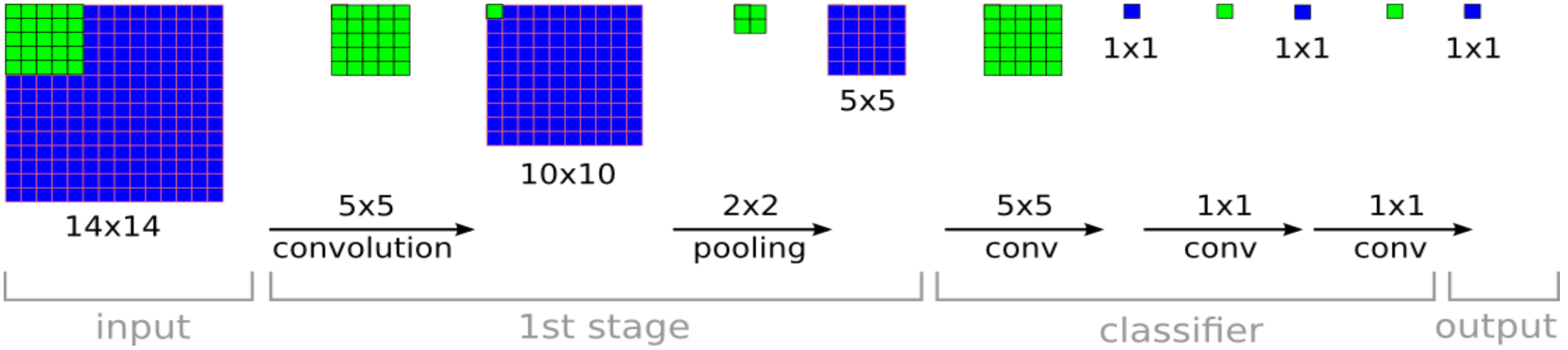


Krizhevksy, Sutskever, Hinton 2012

ImageNet classification breakthrough “SuperVision” CNN

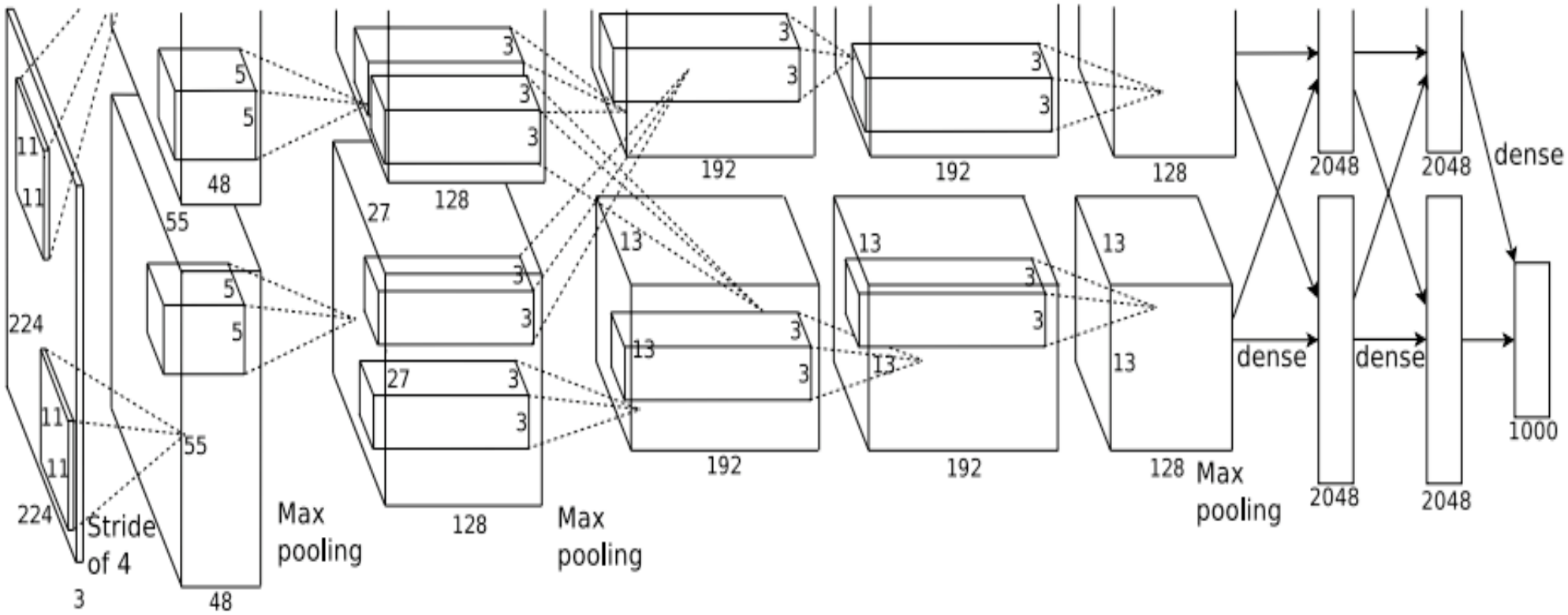
LeNet-5：手写字体识别





[x] LeCun, Y., et al. "Handwritten digit recognition with a back-propagation network." NIPS, 1989.

AlexNet [Krizhevsky'12]



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

卷积网络

- AlexNet 获得了 2012 ImageNet 图像分类赛的最好准确度
- AlexNet的图像分类准确度大幅提升的原因：
  - 层数更深： 8层网络 vs. 3层网络
  - 参数更多：6千万参数 vs. 1百万参数
  - 硬件更快：采用GPU来做训练
  - 数据集更大：1.2百万的图片集和而不是之前的上千大小的数据集
  - 更好的正则化：采用随机丢弃（dropout）技术

卷积网络的新发展

- 新的网络结构不断提出：
  - AlexNet—>NiN—>VGG—>GoogleNet—>ResNet —>DenseNet

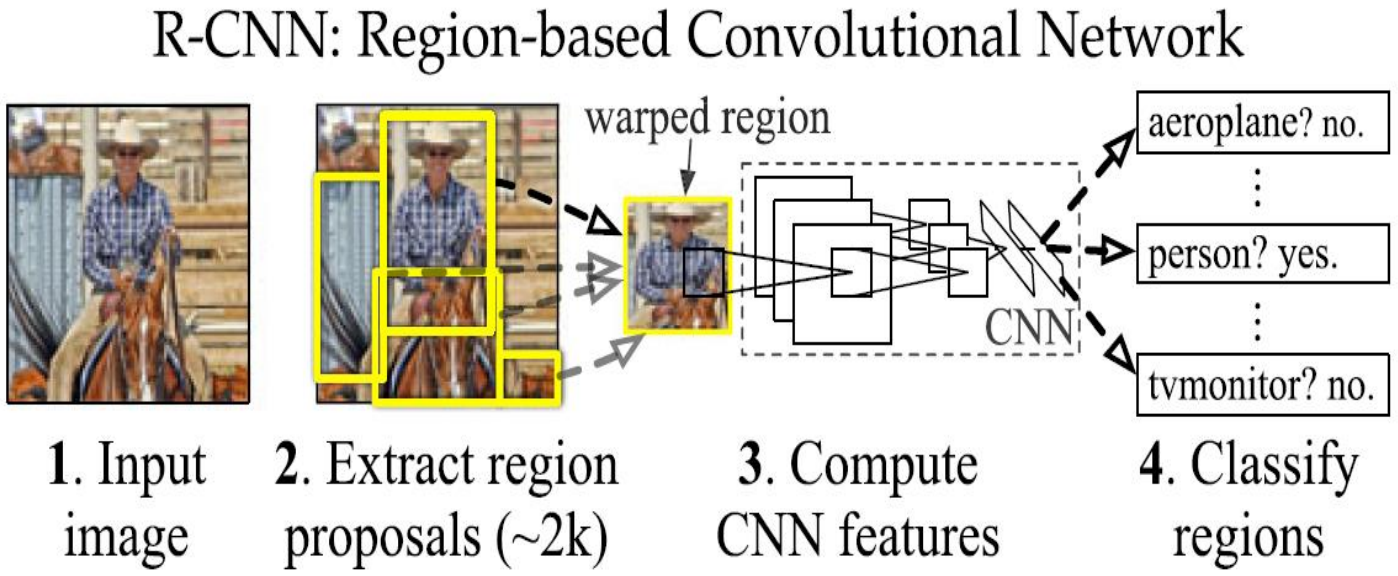
- 网络规模与深度都在增加，准确率提升
  - NiN 引入1*1卷积层（Bottlenecklayer）和全局池化；
  - VGG将7*7卷积核替换成3个3*3卷积核，起到了降参数的作用；
  - GoogLeNet引入了Inception模块；
  - ResNet引入了残差思想，增加了Skip Connection；
  - DenseNet的DenseBlock中将当前层的输出特征，与之后所有的层做直连。

卷积网络

- Densely Connected Convolutional Networks, CVPR 2017. https://arxiv.org/abs/1608.06993.
- SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. https://arxiv.org/abs/1602.07360.
- Xception: Deep Learning with Depthwise Separable Convolutions. https://arxiv.org/abs/1610.02357.
- ResNet: Deep residual learning for image recognition, CVPR 2016. https://arxiv.org/abs/1512.03385.
- GoogLeNet(Inception V3): Going deeper with convolutions, CVPR 2015. https://arxiv.org/abs/1409.4842.
- VGG: Very Deep Convolutional Networks for Large Scale Image Recognition, 2014. https://arxiv.org/abs/1409.1556.
- NiN: Network In Network, 2013. https://arxiv.org/abs/1312.4400.

卷积网络_计算机视觉_对象检测

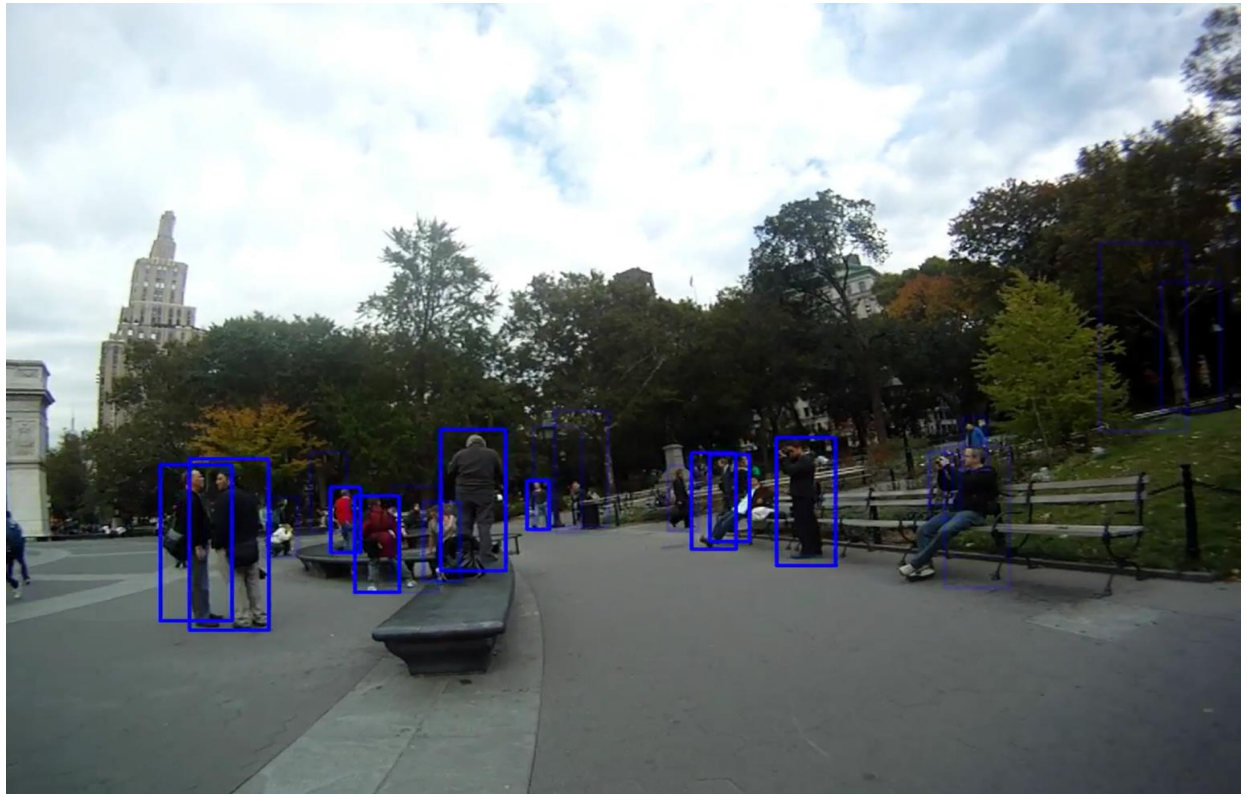
- R-CNN (Region-based Convolutional Network method)
  - Region based convolutional networks for accurate object detection and segmentation, TPAMI, 2015.
  - Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014.



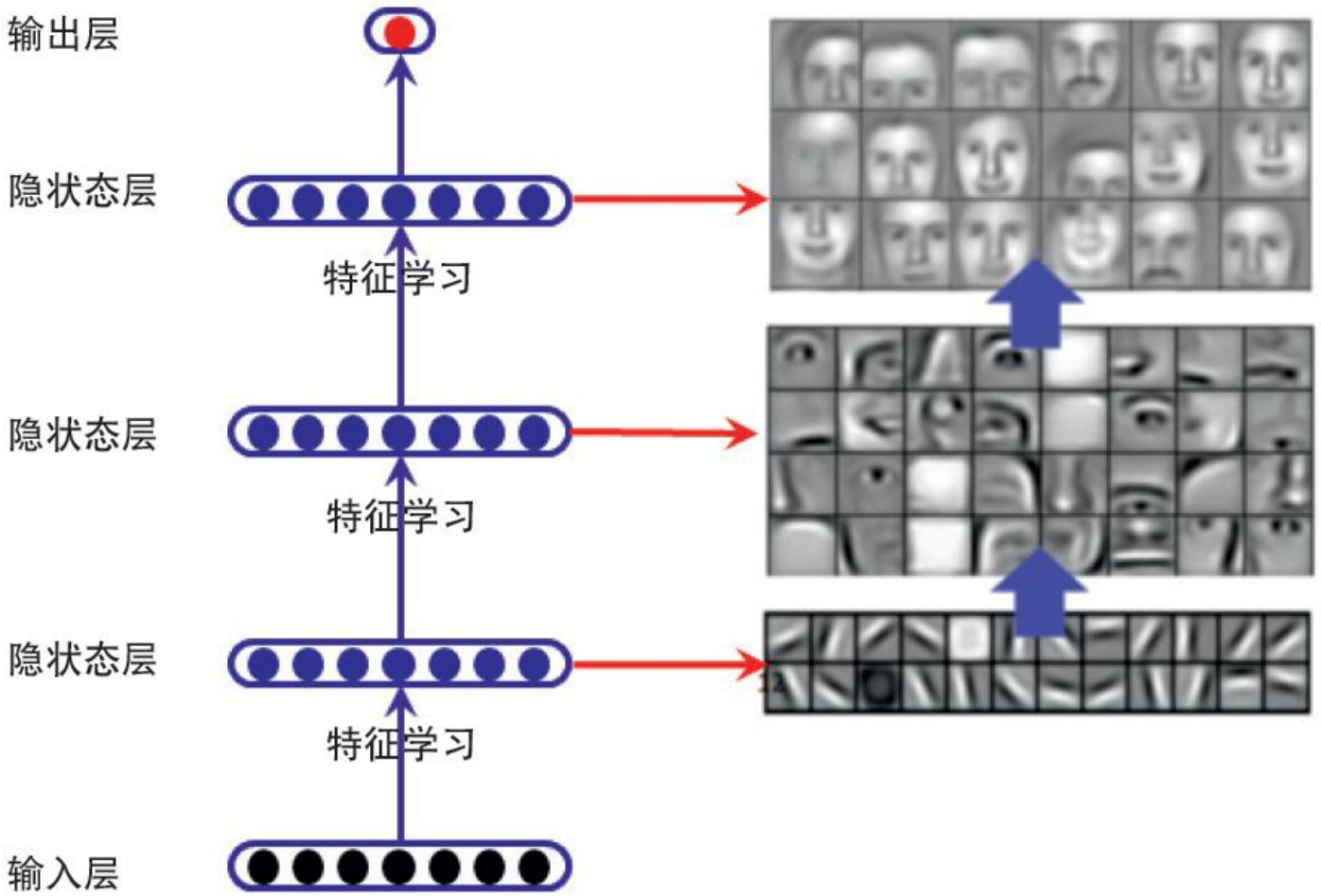
卷积网络_计算机视觉CV_对象检测OD

- [x] Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014.
- [x] Region based convolutional networks for accurate object detection and segmentation, TPAMI, 2015.
- [x] Fast R-CNN, ICCV 2015.
- [x] Faster R-CNN Towards real-time object detection with region proposal networks, NIPS, 2015.
- [x] Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge, IEEE PAMI 2016.
- [x] SSD: Single Shot MultiBox Detector, ECCV 2016.
- [x] You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016.

行人检测(Pedestrian detection)



深度卷积网络-人脸识别



循环网络

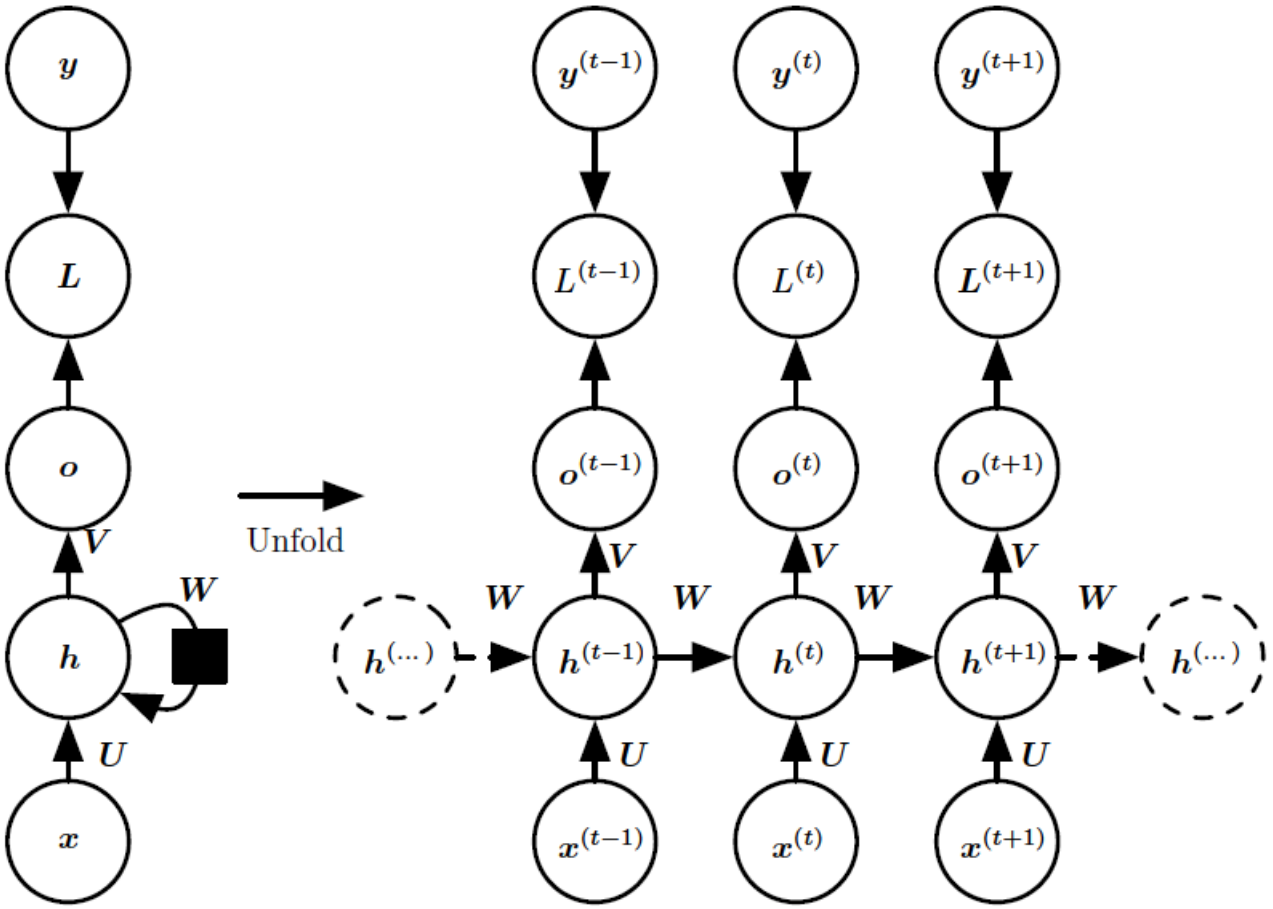
Recurrent network

循环网络-RNN

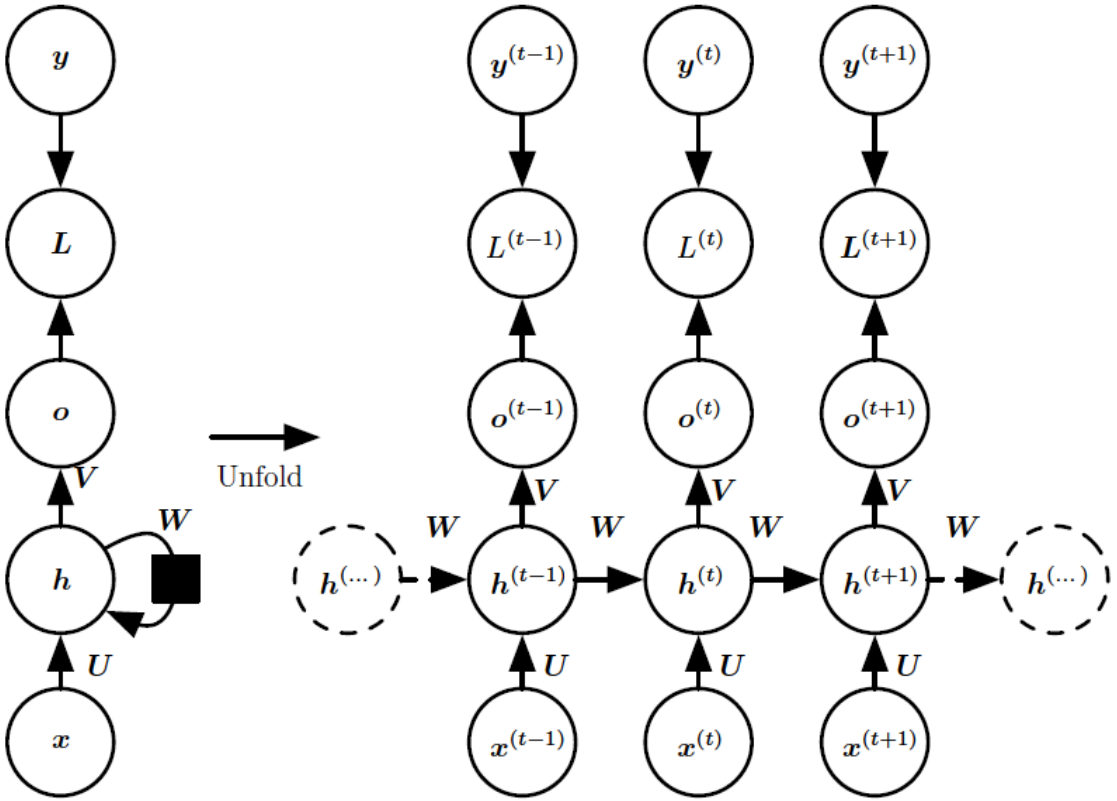
- 循环网络（Recurrent neural network，简称RNN）
- 在时间维度上，每一个时间步处理时，采用共享的权重
- 用于序列建模预测问题：
  - 手写识别（handwriting recognition）、语音识别（speech recognition）
  - 诗歌填词（poem compose）、代码生成（code writing）
  - 股价预测（stock price）、天气预测（weather forecast）
  - 机器翻译（machine translation）、图片注释（image caption）
  - ……
- The Unreasonable Effectiveness of Recurrent Neural Networks, http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

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



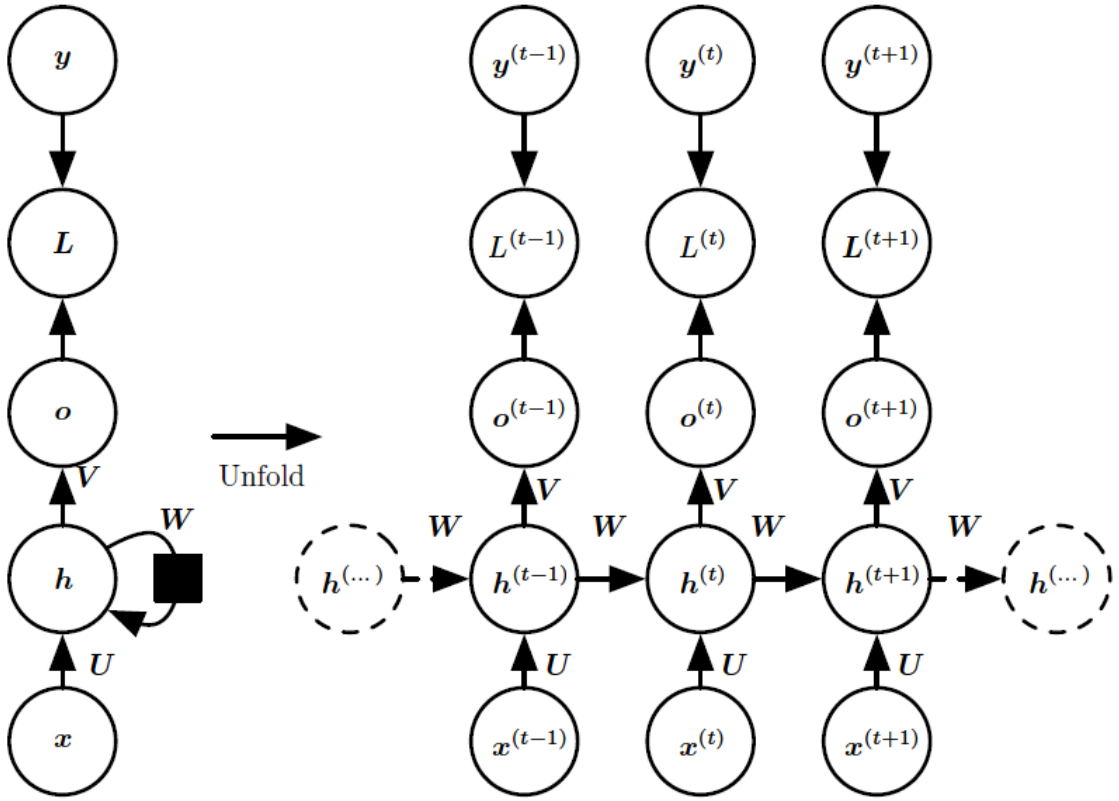
权重共享



- 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

RNN计算图

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)},$$
$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}),$$
$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)},$$
$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)}),$$

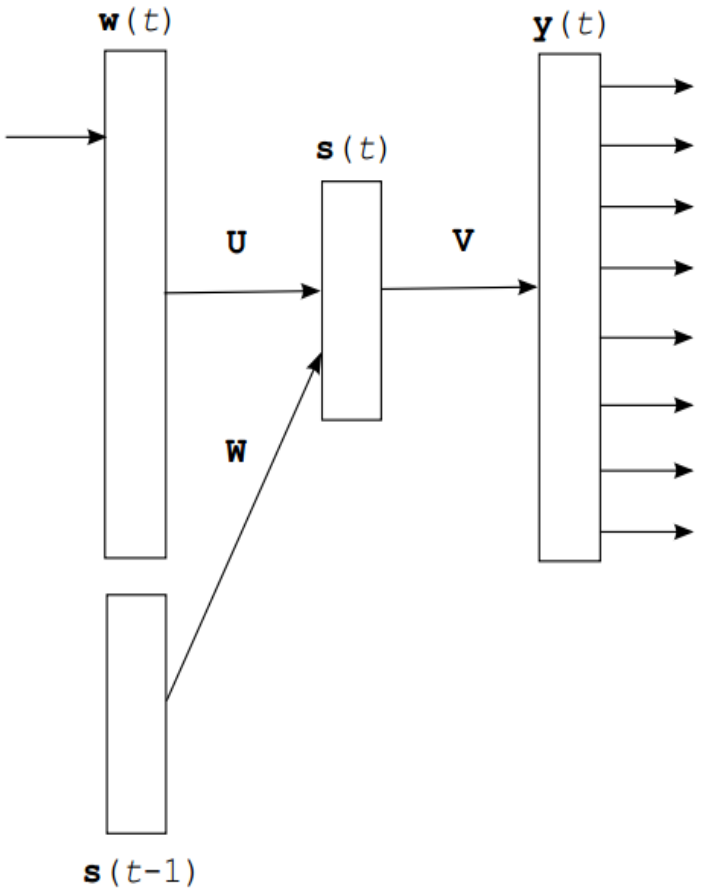


- 循环网络将一个输入序列映射到相同长度的输出序列。
- 信息流动路径：信息在时间上向前（计算输出和损失）和向后（计算梯度）的思想。
- U、V 和W分别对应于输入到隐藏、隐藏到输出和隐藏到隐藏的连接的权重矩阵。
- b 和c 是偏置向量。

simple RNN

h(*t*) = *f*(**U***w(*t*) + **W***h(*t-1*))

y(*t*) = *g*(**V**h(*t*))



f() 是sigmoid函数/tanh函数:

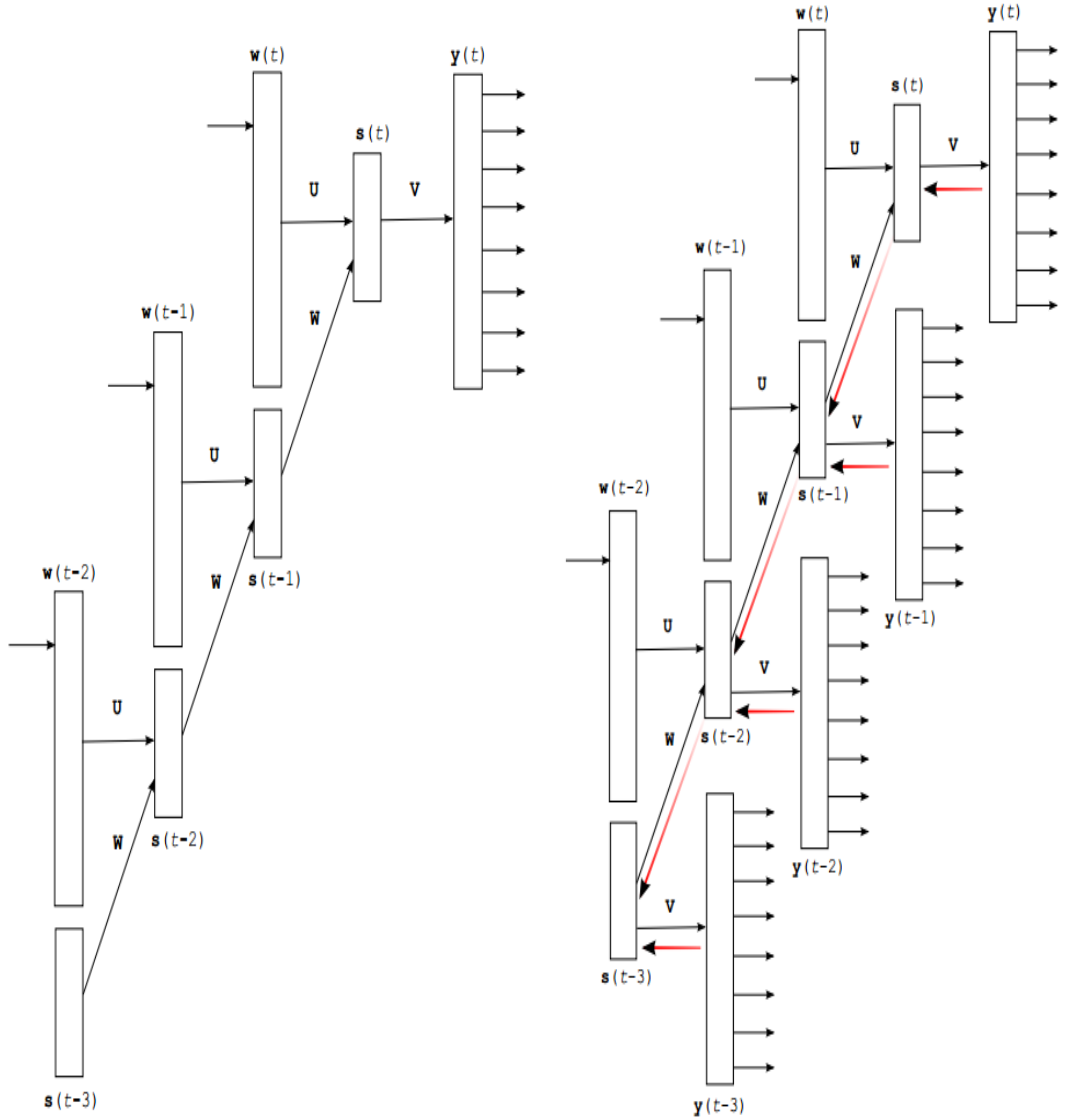
$$\mathrm{f}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad f(z) = \frac{1}{1 + e^{-z}}$$

g() 是 softmax 函数:

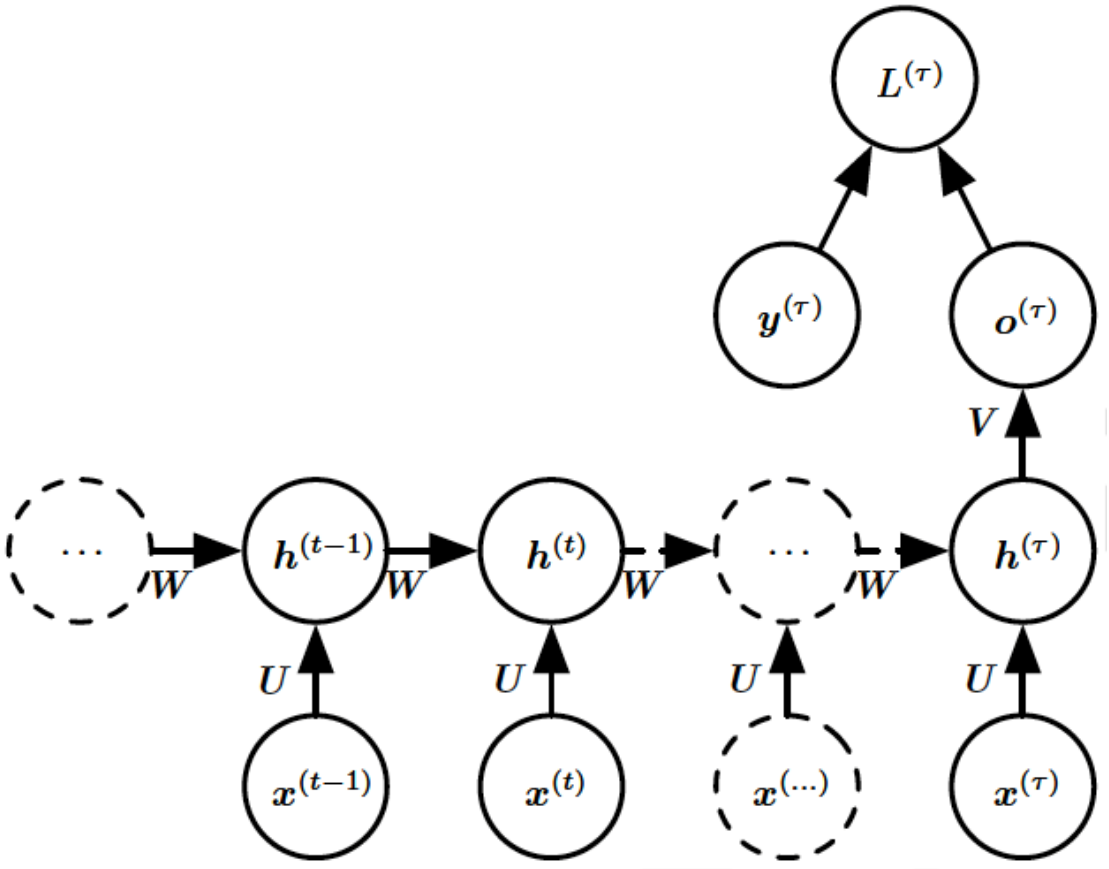
$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

通过时间反向传播（back-propagation through time, BPTT）（截断）

- 在正常的展开的网络（ unfolded ），采用随机梯度下降法进行训练
- 实际中，限制 unfolding steps 为 5 – 10
- 采用批训练来提高计算效率

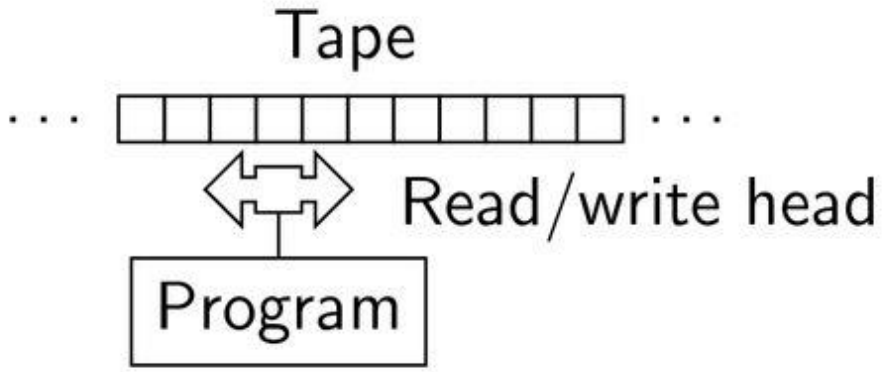


输入与可变输出



- 在时间上展开，在序列结束时具有单个输出。
- 用于概括序列并产生用于进一步处理的固定大小的表示。
- 在结束处存在目标y，或者通过更下游模块的反向传播来获得输出o(t) 上的梯度。
- 应用如：情感分析（sentiment analysis）

图灵机

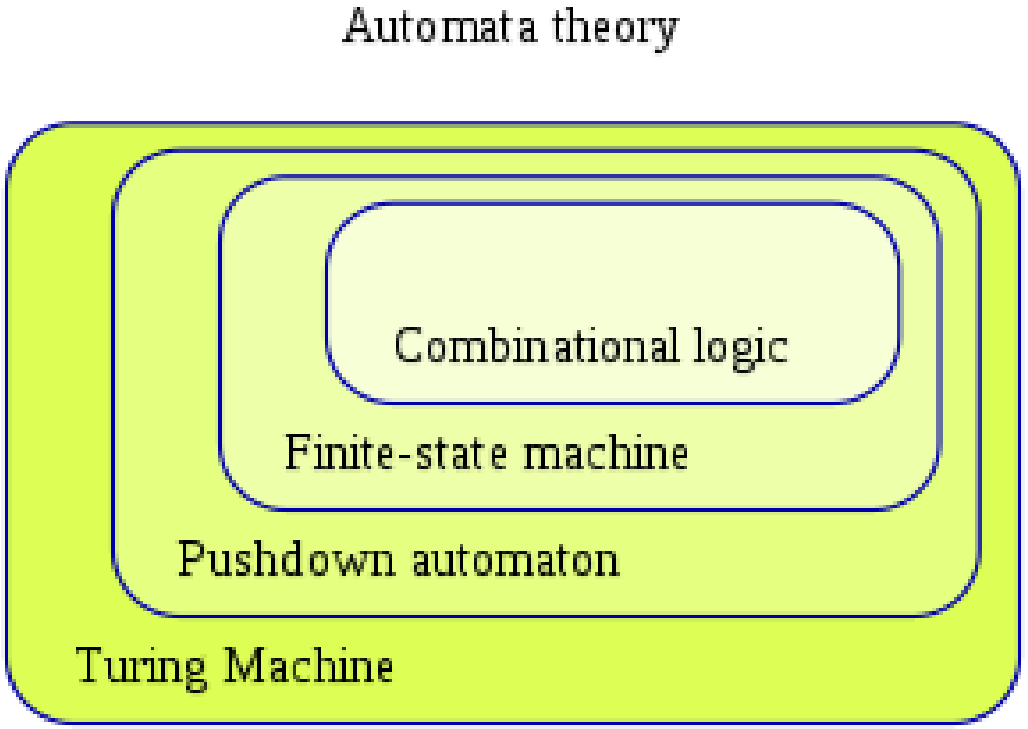


- 英国数学家艾伦·图灵于1936年提出的一种抽象计算模
- **一条无限长的纸带TAPE。**纸带被划分为一个接一个的小格子，每个格子上包含一个来自有限字母表的符号，字母表中有一个特殊的符号 表示空白。纸带上的格子从左到右依次被编号为0, 1, 2, ...，纸带的右端可以无限伸展。
- **一个读写头HEAD。**该读写头可以在纸带上左右移动，它能读出当前所指的格子上的符号，并能改变当前格子上的符号。
- **一套控制规则TABLE。**它根据当前机器所处的状态以及当前读写头所指的格子上的符号来确定读写头下一步的动作，并改变状态寄存器的值，令机器进入一个新的状态，按照以下顺序告知图灵机命令：
  - 1. 写入（替换）或擦出当前符号；2. 移动 HEAD， 'L'向左， 'R'向右或者'N'不移动；
  - 3. 保持当前状态或者转到另一状态
- 一个状态寄存器。它用来保存图灵机当前所处的状态。图灵机的所有可能状态的数目是有限的，并且有一个特殊的状态，称为**停机状态。**

https://zh.wikipedia.org/wiki/图灵机

图灵机

- 自动机理论
  - 组合逻辑电路
  - 有限自动机
  - 下推自动机
  - 图灵机



[x] Michael Sipser. Introduction to the Theory of Computation. PWS Publishing, 1997.
[x] Lewis, Harry R., and Christos H. Papadimitriou. Elements of the Theory of Computation. Prentice Hall PTR, 1997.

循环网络的近似能力

- Schäfer 和Zimmermann的论文证明（ICANN-2006）：

用Sigmoid激活函数的RNN是图灵完备的（Turing-complete），即只要给出正确的权重，RNNs可以计算任何可计算的程序。

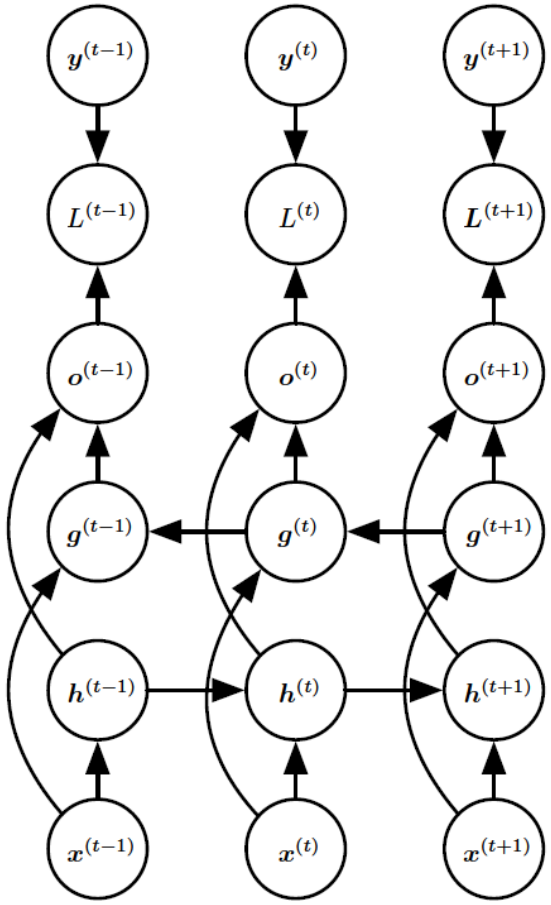
- RNN是图灵完全等价的 (Siegelmann and Sontag, 1995)

任何图灵可计算的函数都可以通过一个有限维的循环网络计算。

RNN 经过若干时间步后读取输出，这与由图灵机所用的时间步是渐近线性的，与输入长度也是渐近线性。

[x] Schäfer, Anton Maximilian, and Hans Georg Zimmermann. "Recurrent neural networks are universal approximators." International Conference on Artificial Neural Networks (ICANN). Springer, Berlin, Heidelberg, 2006.
[x] Siegelmann, Hava T., and Eduardo D. Sontag. "On the computational power of neural nets." Proceedings of the fifth annual workshop on Computational learning theory. ACM, 1992.

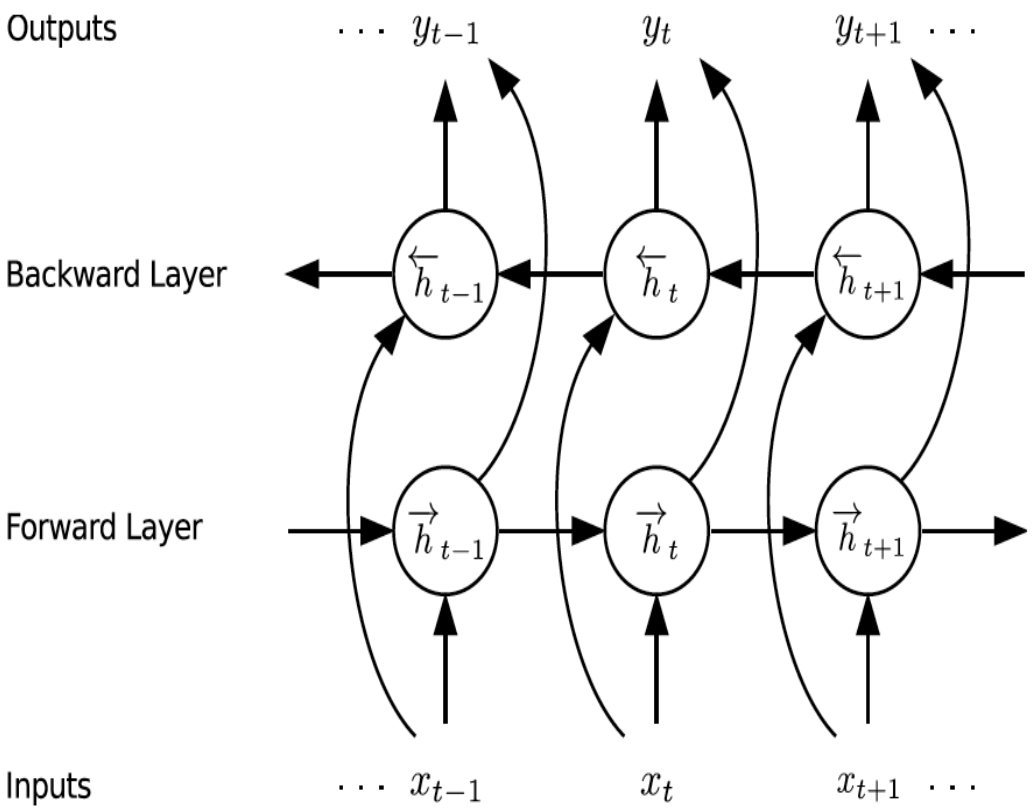
双向RNN（Bidirectional RNN）



- 双向RNN，结合时间上从序列起点开始移动的RNN 和另一个时间上从序列末尾开始移动的RNN。
- 双向RNN，其中h(t) 代表通过时间向前移动的子RNN 的状态，g(t) 代表通过时间向后移动的子RNN 的状态。
- 输出单元o(t)，能够计算同时依赖于过去和未来且对时刻t的输入值最敏感的表示
- 用于手写识别和语音识别

[x] Alex Graves et al., Speech recognition with deep recurrent neural networks, ICASSP 2013.

双向RNN



- 更新方程

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y}\overrightarrow{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y$$

损失函数

- 循环网络的训练损失：
  - 将*x* 值的输入序列映射到输出值o 的对应序列。
  - 损失L 衡量每个输出o 与相应的训练目标y 的距离。
  - 使用softmax 输出时，o 是未归一化的对数概率（logit）。
  - 损失L 内部计算：y’ = softmax(o)，并将其与目标y 比较。

损失函数

- 与x 序列配对的y 的总损失就是所有时间步的损失之和。例如，L(t) 为给定的x(1), ⋯ , x(t) 后y(t) 的负对数似然

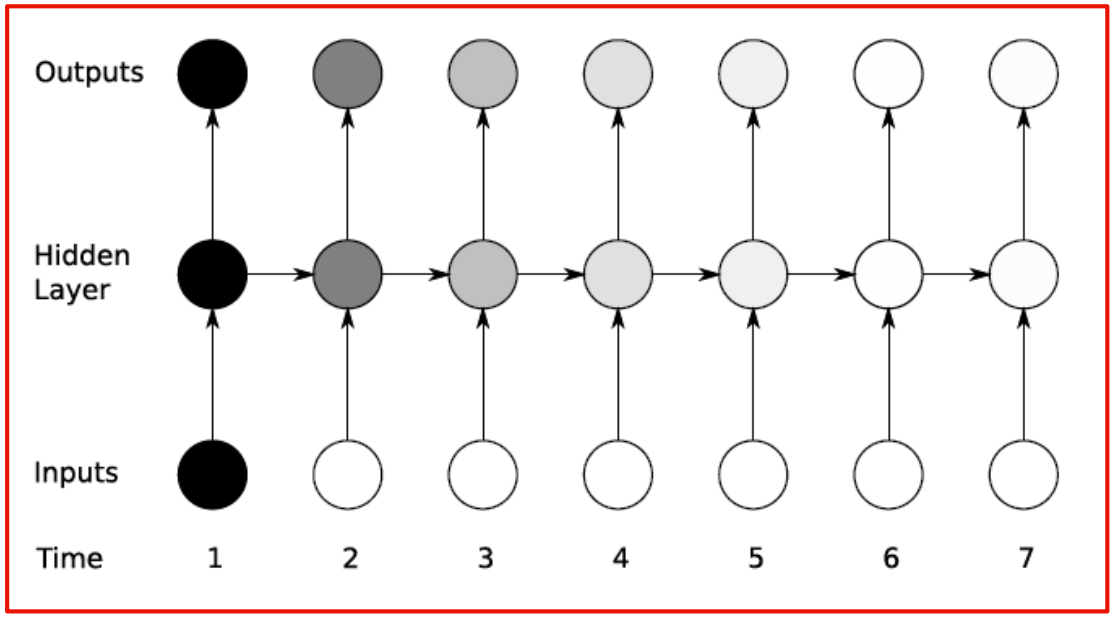
$$
\begin{aligned}
&L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right) \\
&= \sum_t L^{(t)} \\
&= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right)
\end{aligned}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

通过时间反向传播（back-propagation through time, BPTT）

- 通过时间反向传播算法（back-propagation through time, BPTT）应用于展开图的反向传播参数调节。
- 梯度计算涉及执行一次前向传播（从左到右的传播），一次由右到左的反向传播。
- 运行时间是O(T)，并且不能通过并行化来降低，因为前向传播图是固有顺序的。
- 每个时间步只能一前一后地计算。前向传播中的各个状态必须保存，直到它们反向传播中被再次使用，因此内存代价也是O(T)。
- 损失函数的梯度关于各个参数的计算是计算成本很高的操作。
- 由此可见，循环网络非常强大但训练代价也很大。

RNN训练的问题

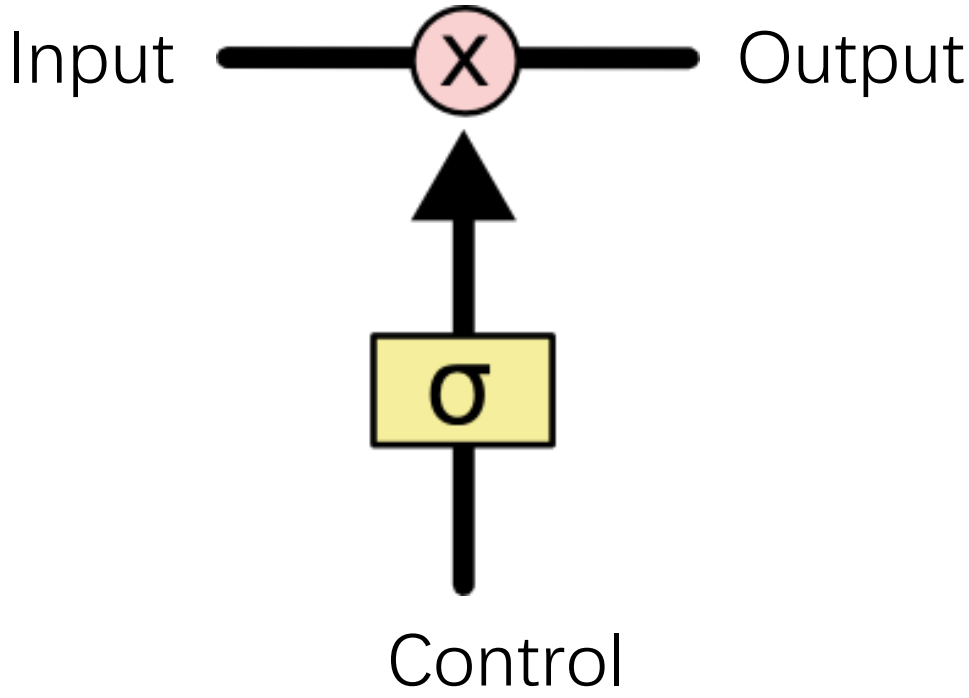
- Gradient Clipping
  - 训练循环网络时，经常出现梯度要么太大，要么太小，为了加速训练，需要把梯度设置为一些固定数值。比如说，梯度的任何维度的数值应该小于1，如果某个维度的数值大于1，则固定设置为1.
- 训练RNN时的梯度消失和梯度爆炸问题
  - BPTT时，最初和最后的时间步的梯度的幅度过大或者过小
  - Sigmoid函数的饱和问题
  - ReLU函数的问题
- LSTM解决以上问题



[1] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A Field Guide to Dynamical Recurrent Neural Networks, 2001.

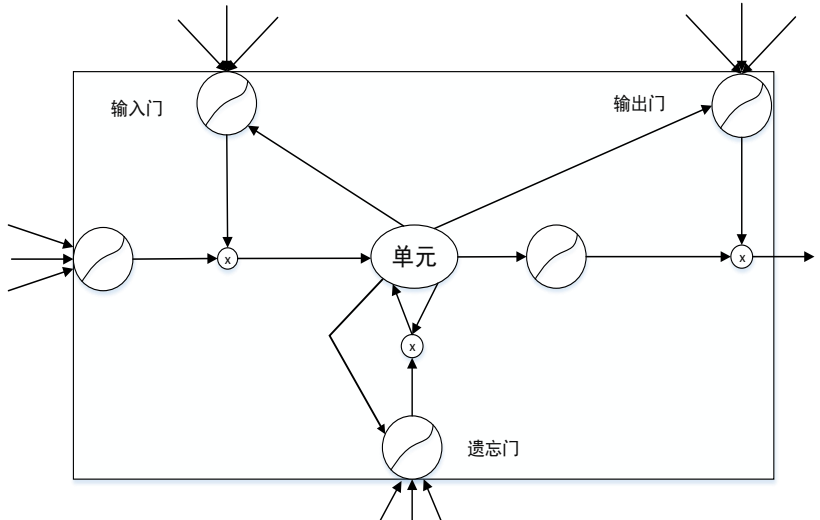
门电路

- 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



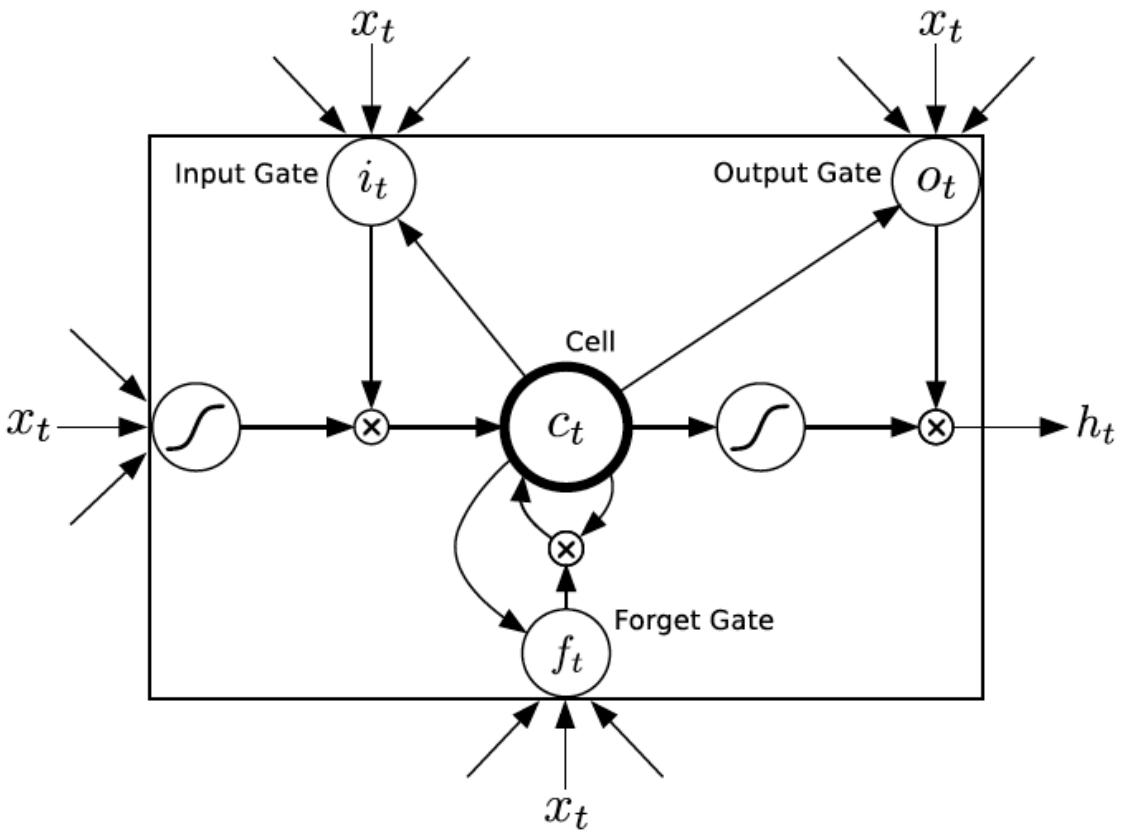
长短时记忆网络（Long-short term memory ）

- 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:
  - 记忆单元（memory cell）、输入门（input gate）、遗忘门（forget gate）及输出门（output gate）。
- 三个辅助输入分别控制网络是否输入，是否存储输入以及是否输出。
  - 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。



https://distill.pub/2019/memorization-in-rnns/

LSTM



- 前向方程

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

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

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

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

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

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

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

- RNN

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

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

- LSTM

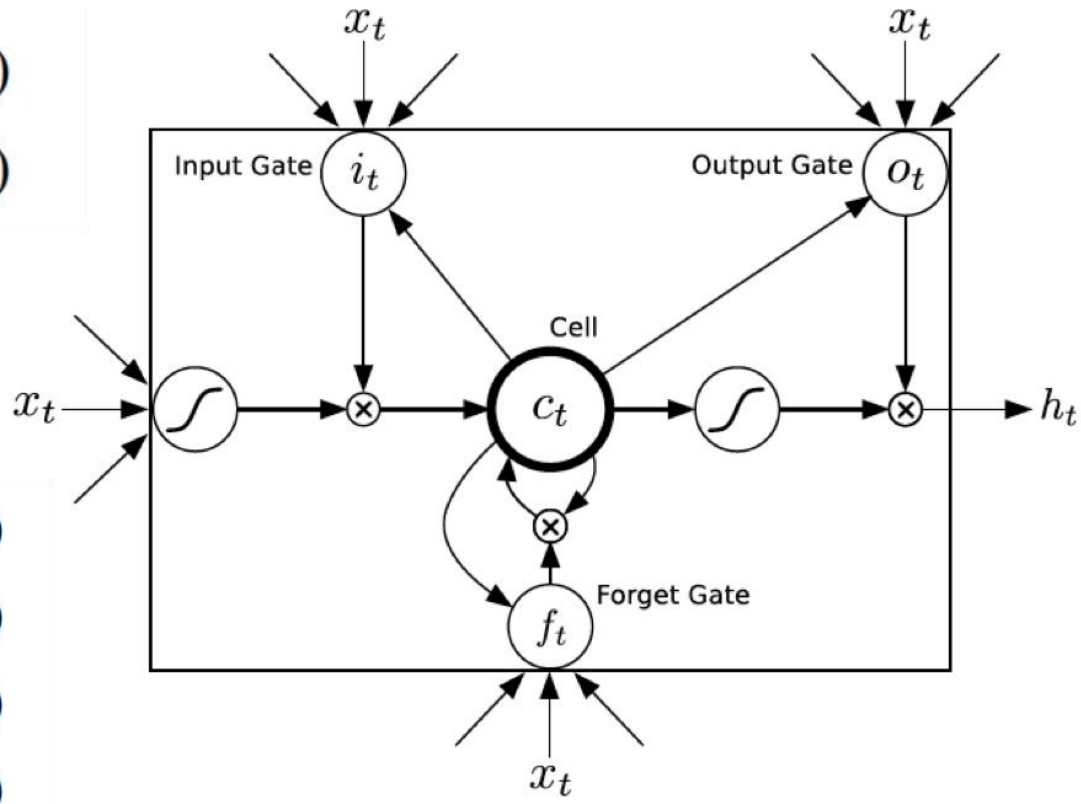
$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \quad (3)$$

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

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

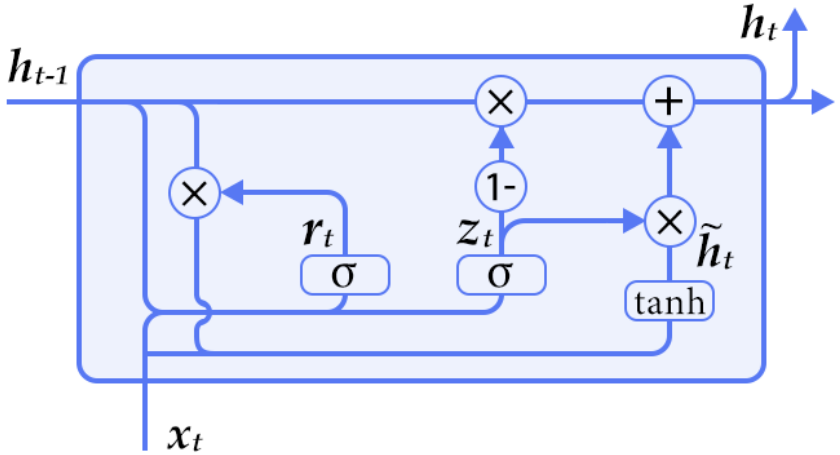
$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \quad (6)$$

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



门控循环单元（Gated Recurrent Unit）

- GRU （Gated Recurrent Unit）是Cho等提出的LSTM的简化版本，也是RNN的一种变种
- GRU单元只有两个门：
  - 更新门（update gate），将LSTM的输入门和遗忘门合并，用于控制历史信息对当前时刻隐层输出的影响。如果更新门接近1，会把历史信息传递下去。
  - 重置门（reset gate），如果重置门关闭，会忽略掉历史信息，即历史不相干的信息不会影响未来的输出。



$$z_t = \sigma(W_t \cdot [h_{t-1}, x_t]) \quad \text{更新门}$$

$$r_t = \sigma(W_t \cdot [h_{t-1}, x_t]) \quad \text{重置门}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \quad \text{节点状态}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad \text{输出}$$

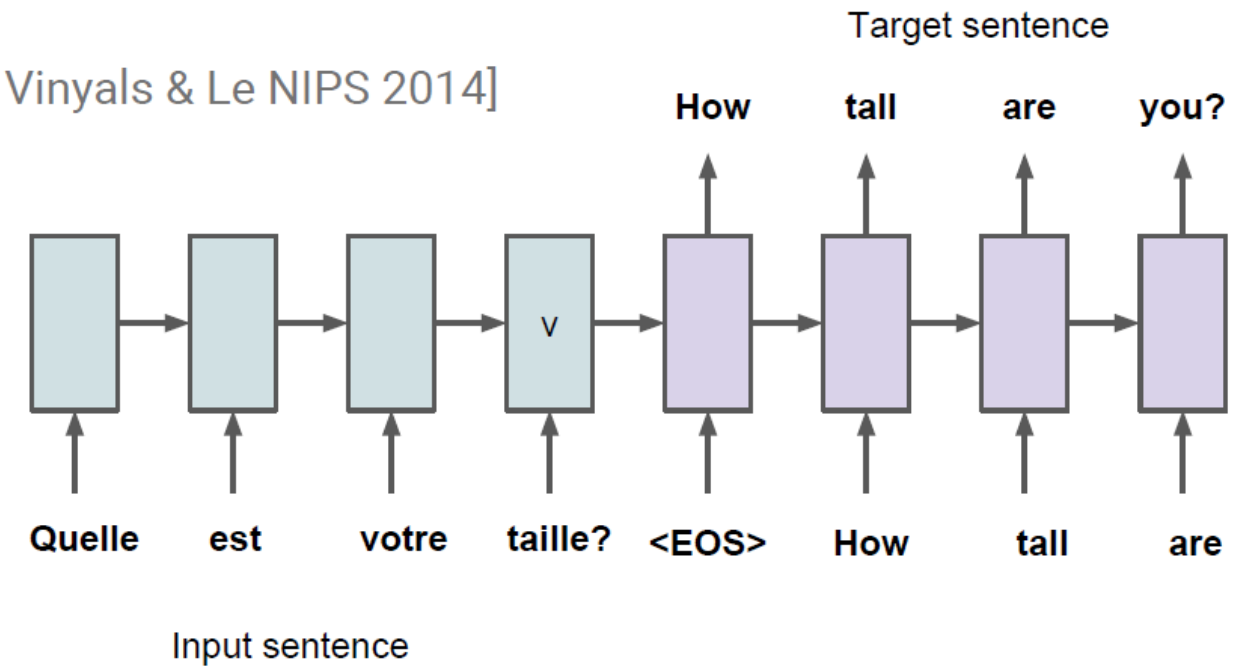
[x] K. Cho et al., On the Properties of Neural Machine Translation Encoder-Decoder Approaches, SSST-8, 2014.

序列对序列模型

Sequence-to-Sequence Models

机器翻译（英语-法语）

[Sutskever & Vinyals & Le NIPS 2014]



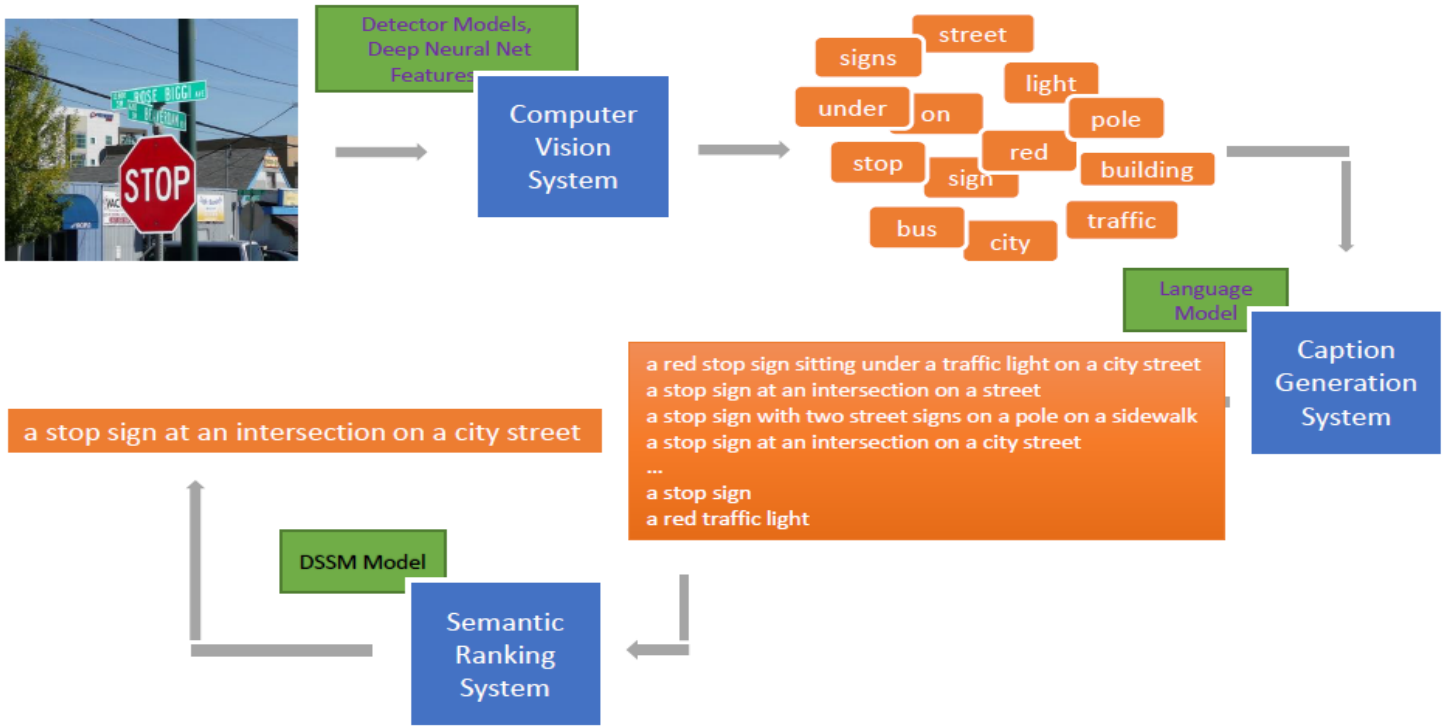
[x] I. Sutskever, O. Vinyals, & Q. V. Le, Sequence to sequence learning with neural networks. NIPS 2014.

注意力机制

- 序列对序列模型Seq2Seq是一个万能模型

- 注意力机制（ attention mechanism ）

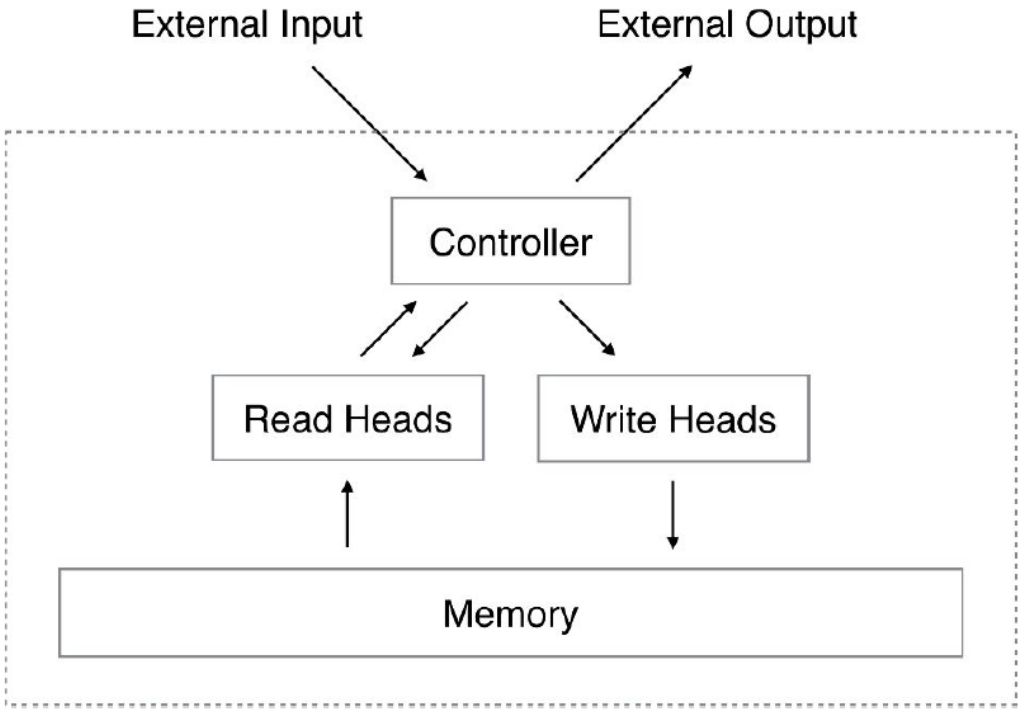
深度学习的应用-图片注解



Fang, Hao, et al. "From captions to visual concepts and back." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

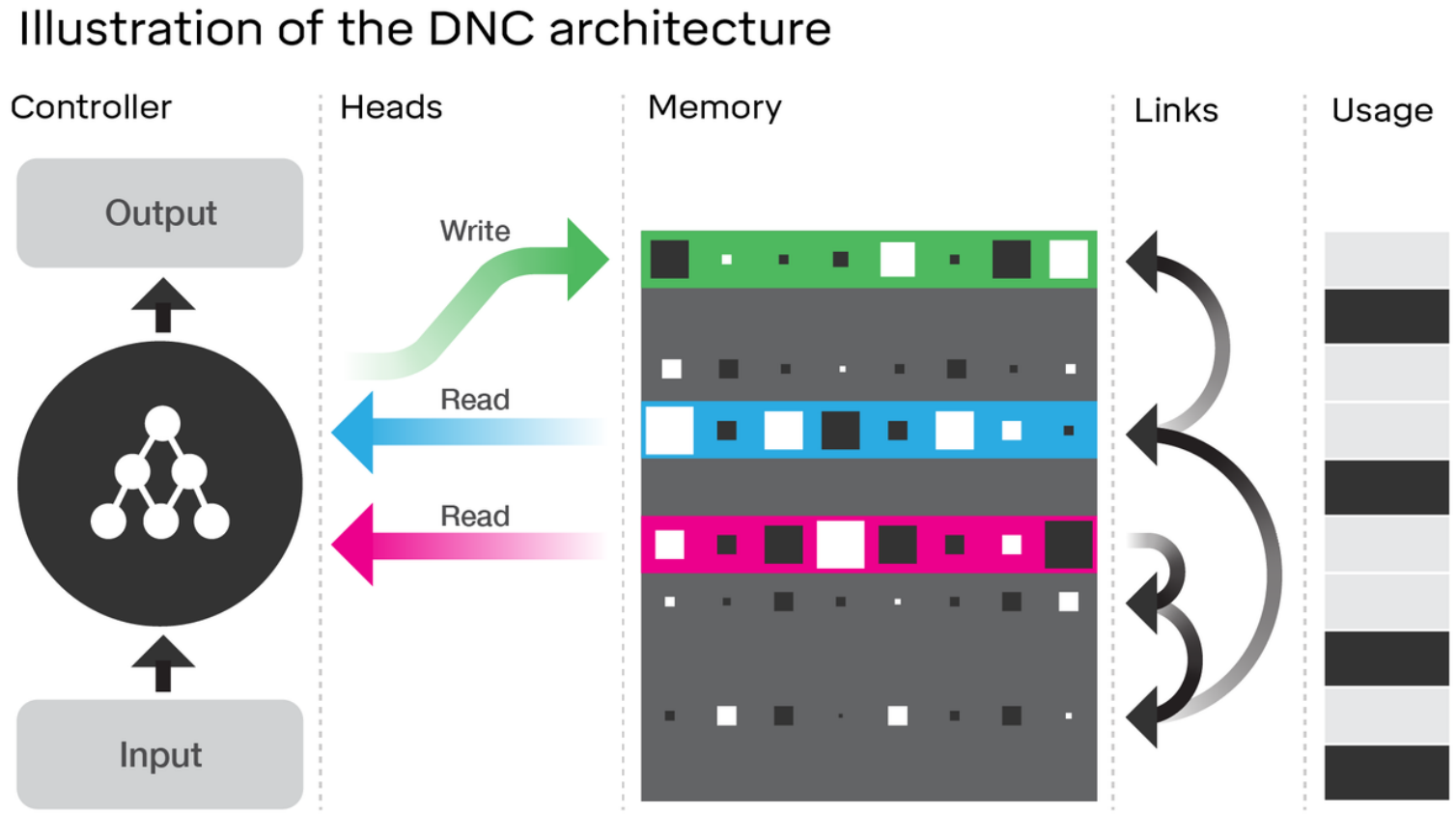
神经图灵机（Neural Turing Machines）

- NTM组件
  - 控制器
  - 外部记忆
  - 读写操作
  - 输入输出
- 整个架构可微分



[x] Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." *arXiv preprint arXiv:1410.5401* (2014). http://arxiv.org/abs/1410.5401

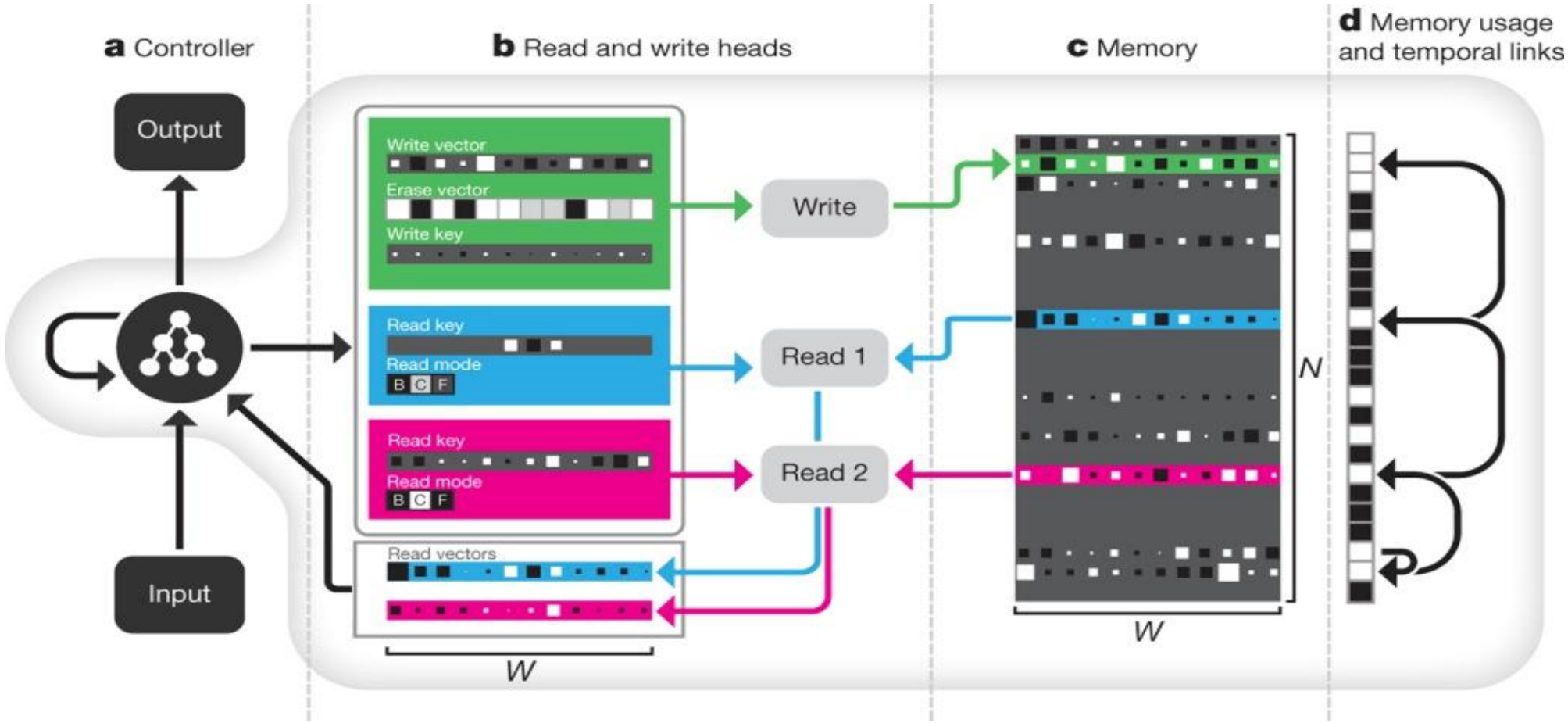
Differentiable neural computers (DNC)



[x] Graves, Alex, et al. "Hybrid computing using a neural network with dynamic external memory." Nature 538.7626 (2016): 471.

DNC architecture

nature



Alex Graves, New Directions For Recurrent Neural Networks, invited talk, ICLR 2017.

ASR-paper参考

- [x] Long short term memory neural computation, Neural computation 9 (8), 1735-1780, 1997.
- [x] Connectionist temporal classification labelling unsegmented sequence data with recurrent neural networks, ICML 2006.
- [x] Speech recognition with deep recurrent neural networks, ICASSP 2013.
- [x] Towards End-To-End Speech Recognition with Recurrent Neural Networks, ICML 2014.
- [x] Speech 2 End-to-End Speech Recognition in English and Mandarin, JMLR 2016.
- [x] Yoshua Bengio, Ian Goodfellow, Aaron Courville, Deep Learning, MIT Press, 2016.

谢谢指正！

zhenchen@Tsinghua.edu.cn