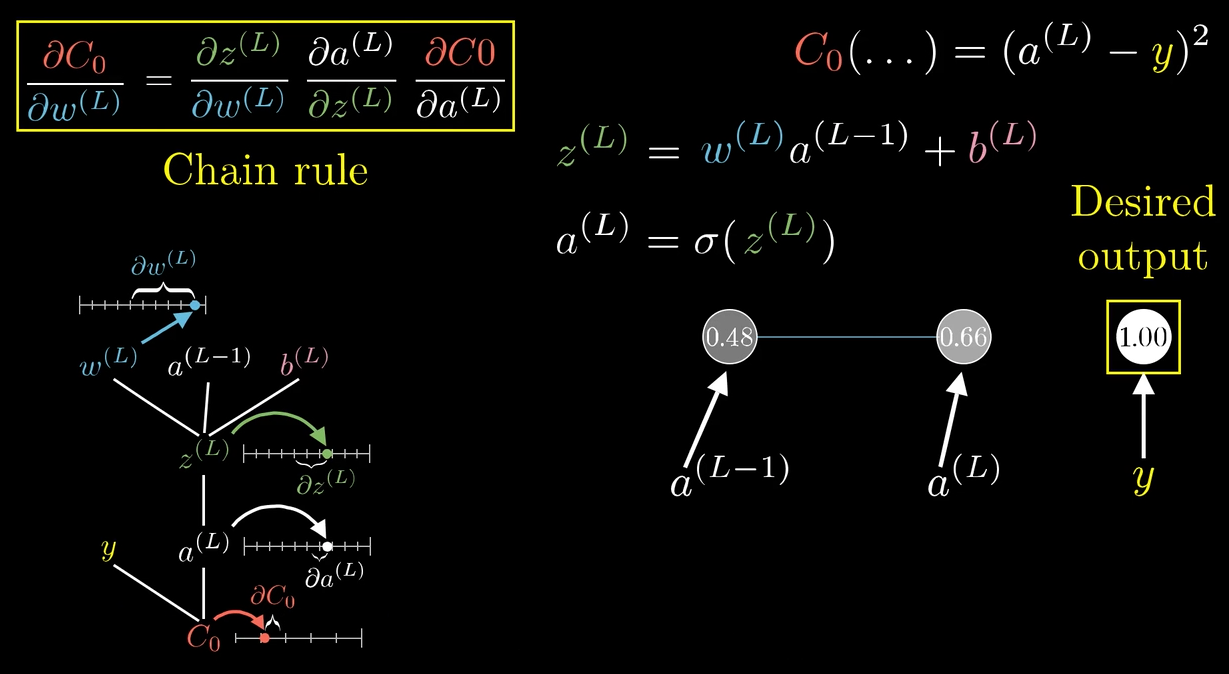
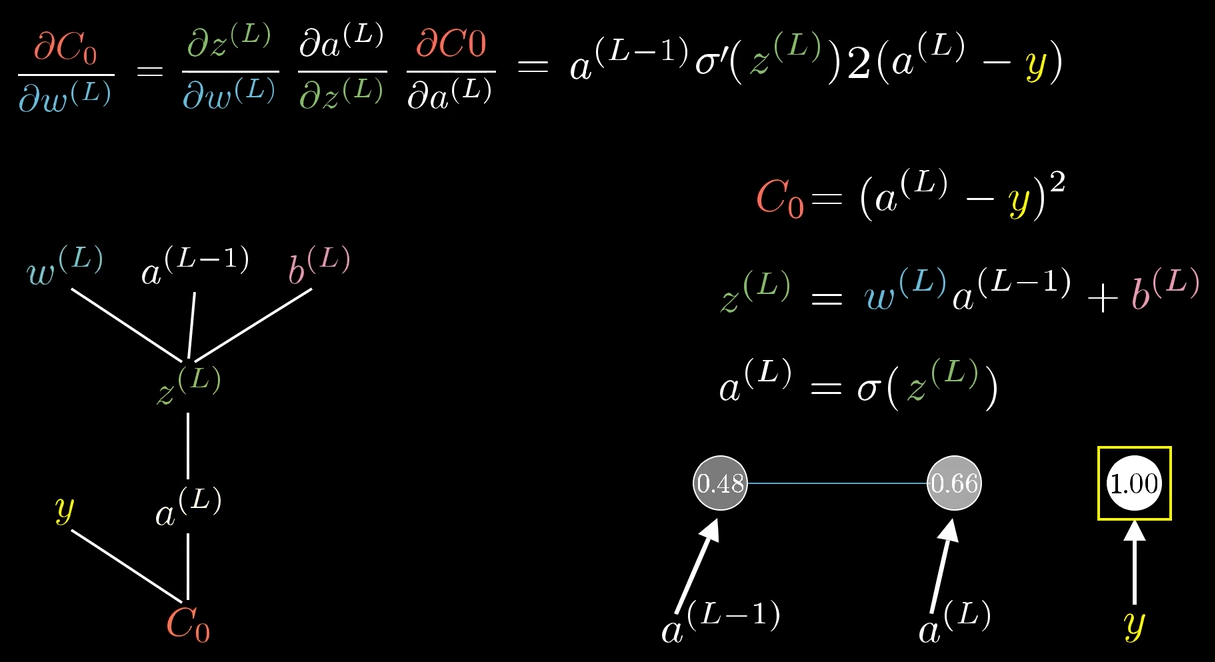
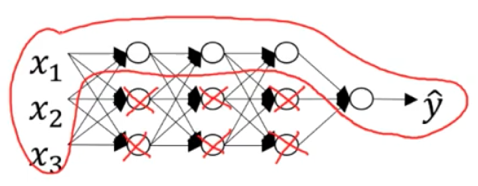
* X(n\*m)，n个特征，m个样本；y(1\*m)。
* 参数θT = {b, w1, w2, w3, …}，共n+1个参数。
* 神经网络层数计算不包括输入层。
* 表示方法：。
* 激活函数用于引入非线性因素，解决线性不能解决的问题。
* 正向传播：第L层神经元的值由第L层的(w、b)与第L-1层的神经元值加权后输入激励函数计算得出。
* 反向传播：根据输出层产生的误差依次从后向前更新隐藏层的参数w、b。

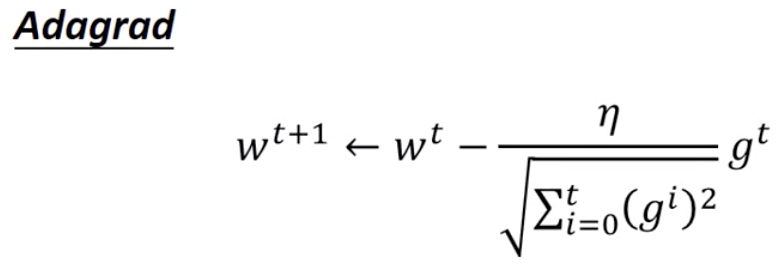


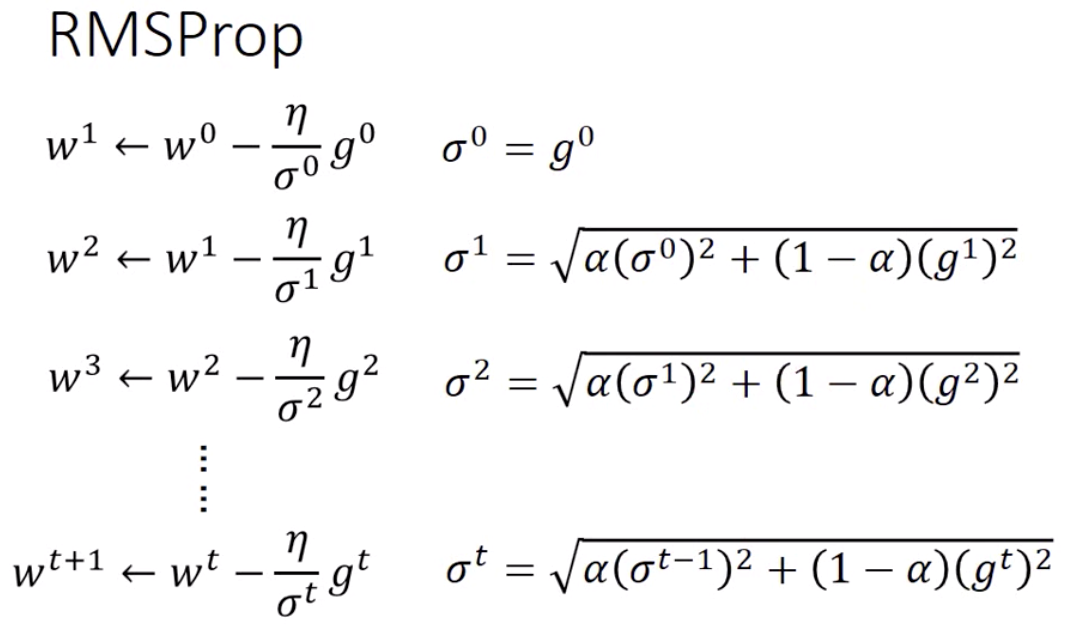


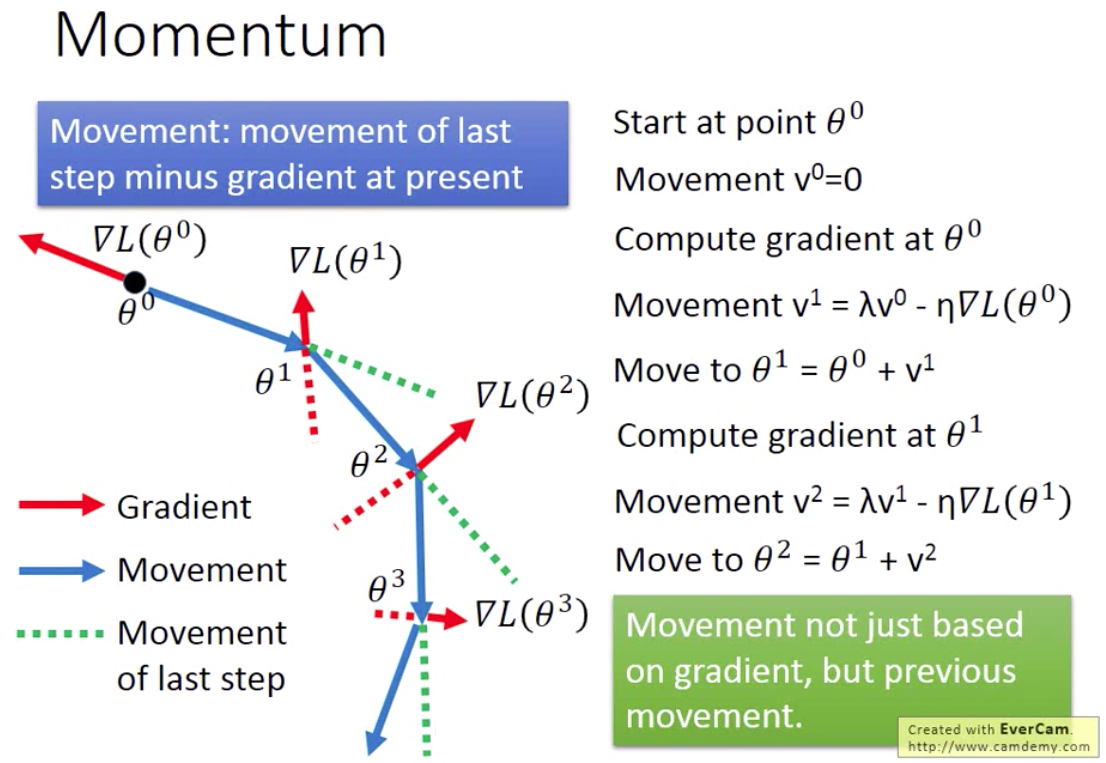
* sigmoid激活函数输出平均值不为0需要平移，计算复杂，导数梯度小，学习速率慢，很容易产生梯度消失。
* tanh激活函数经过零点，平均值更接近0，不需要更多的平移，可以使下一层学习更方便，但计算复杂，容易产生梯度消失。
* 对于输出层，二分类时使用sigmoid，多分类时使用softmax。
* 输入层神经元等于特征个数，输出层神经元等于类别个数。
* 当函数z值较大或较小时，sigmoid和tanh导数梯度很小接近于0，导致计算很慢和梯度消失，此时relu更好，relu在零点不可导，但实际计算中z=0时很少出现。
* w若初始化为0，每一层神经元输出值相同，梯度下降后每个神经元都是对称的，相当于在训练同一个函数；w若初始化较大，会导致函数z值较大，sigmoid/tanh梯度会很小；因此w应初始化为较小的随机数，b可以初始化为0。
* Xavier-tanh：W = np.random.randn(in,out)/np.sqrt(in)。
* He-relu：W = np.random.randn(in,out)/np.sqrt(in/2)。
* 深度神经网络低层学习简单的特征，高层学习复杂的特征。
* 超参数：网络层数、神经元个数、学习率、正则化、优化器、权重初始化方式、迭代次数、激活函数、损失函数、小批量数据。
* L2正则化，当正则化系数λ无限大时，w[L]≈0，可以看作深度很大的逻辑回归，从而抑制了过拟合。



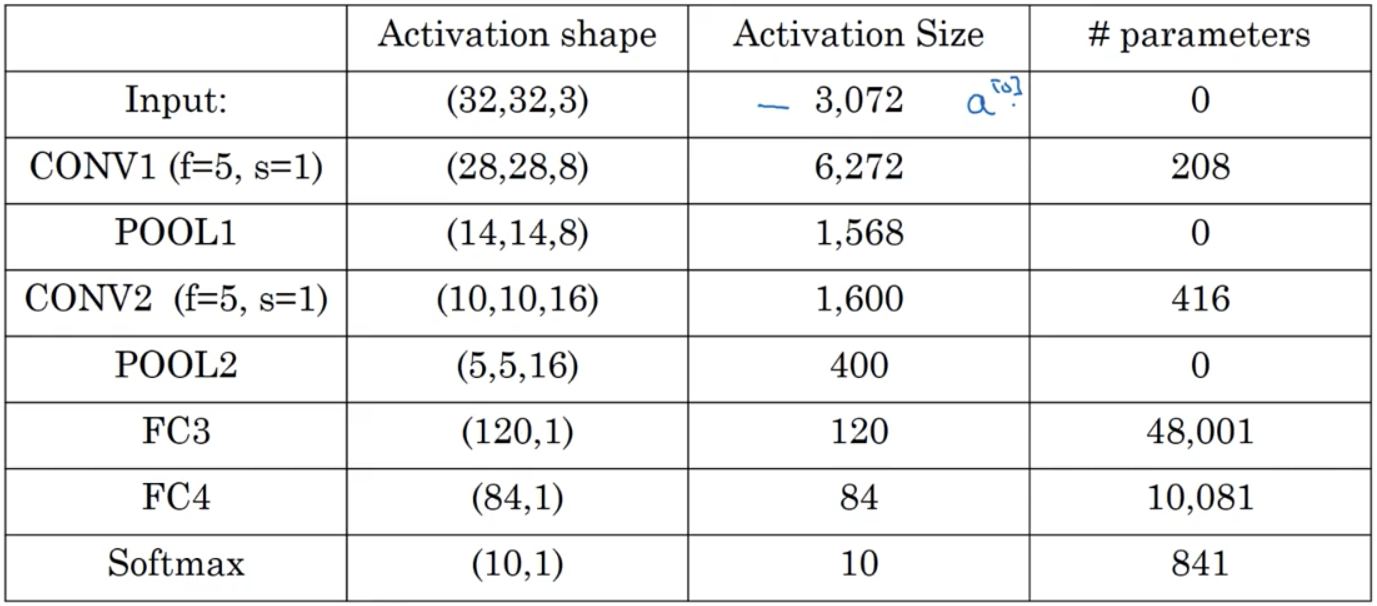
* dropout正则化，一种集成学习，随机保留该层网络dropout的神经元进行训练，从而抑制过拟合，不同层dropout可以选择不同取值。若训练时使用dropout=p%，测试时w=w\*(1-p)%。
* 交叉熵损失函数。
* 梯度爆炸和梯度消失出现在反向传播过程中。
* 梯度爆炸：激活函数求导大于1，导致学习参数越来越大，(1)隐藏层的层数过多；(2)权重初始化过大；(3)学习率太大。
* 梯度消失：激活函数求导小于1，导致学习参数越来越小，(1)隐藏层的层数过多；(2)激活函数不合适，如sigmoid只在[-5,5]存在导数，其他大部分导数趋于0。
* 解决梯度问题：(1)减少网络层数；(2)使用relu/leak-relu激活函数(导数≡1)；(3)合理初始化权值如He初始化；(4)使用LSTM。
* SGD：总数m个训练样本，每次循环训练n个，完整训练一次m个样本需要m/n次循环，每次循环中对n个样本进行：(1)前向传播计算z、a；(2)计算成本函数；(3)反向传播计算梯度dw、db；(4)更新、。在样本量很大的情况下，使用batch完整训练一次只使用一次梯度下降，使用mini-batch完整训练一次使用m/n次梯度下降，运行更快更准确；每个mini-batch梯度下降不一定都会向全局最小值靠近，可能会寻找局部最小值的方向，会走一些弯路(噪音)，但总体趋势是全局最小值的方向；随时间增长逐渐减小学习率有助于更好的找到最小值。
* Momentum：计算梯度的加权平均数，使梯度摆动幅度减小，并利用该梯度更新w、b；，，，，β一般取值0.9。
* RMSProp：计算梯度平方的加权平均数，并利用该梯度更新w、b，使梯度摆动幅度减小，可以加快收敛速度；，，，，β一般取值0.999，ε一般取值10-8。
* Adam：结合Momentum和RMSProp；(1)参数更新：，，，；(2)偏差修正：，，，；(3)更新w、b：，b；β1一般取值0.9，β2一般取值0.999，ε一般取值10-8。
* 图例：



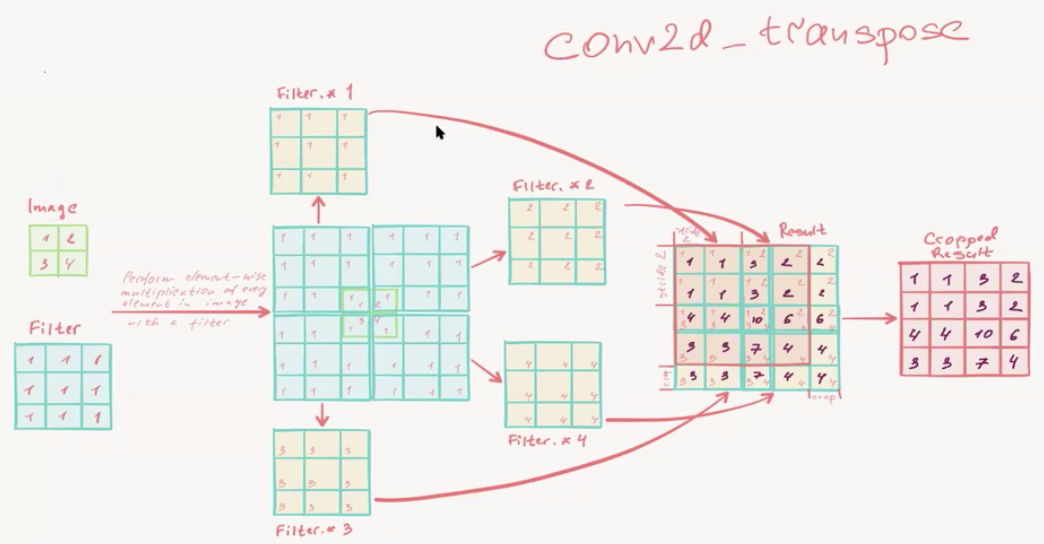




* BatchNorm(BN)：对输入特征归一化后输入第一层网络，之后每层网络输出的z值分布一直在变化，为了下一层网络能更好的训练，要对每一层网络输出的z进行batch归一化，在输入激活函数前执行。BN使每层的输入值变得更稳定，减弱了前层参数作用与后层参数作用的关系，有助于加速整个网络的学习，还有轻微正则化的效果。
* softmax若要分成n(n>2)类，则输出层用n个神经元，每个神经元代表一个种类的概率，。
* 迁移学习：寻找一个相关问题并且已经训练好的神经网络，删除网络的输出层和输入输出层的参数w[L]、b[L],用新的数据集进行训练，若数据集小则只训练输入输出层的参数，若数据集大可以训练更多层的参数。
* 多任务学习：一个数据有多个标签，神经网络对一个数据样本针对多个问题分别预测不同结果，如识别一张图片上是否有行人、汽车、交通灯等，多应用于计算机视觉物体检测。
* 卷积神经网络--------------------------------------------
* 二维卷积：步长为s，(1)图片尺寸缩小padding=0，输入(n\*n)，过滤器(f\*f)，输出；(2)图片尺寸不变padding>0，输入(n\*n)，过滤器(f\*f)，边缘填补p层，输出，s=1时，，因此f必须为奇数才能保证p是整数。注：仅s=1时，padding=same输出相同尺寸。
* 三维卷积：p=0，s=1，输入(n\*n\*nc），过滤器(f\*f\*nc)，输出[(n-f+1)\*(n-f+1)\*nci]，nc为通道数/深度，输入的深度和过滤器的深度必须相等，输出的深度取决于使用过滤器卷积的次数。
* ，，  
  ，，  
  ，  
  ，，  
  ，。
* 池化：(1)max pooling，用过滤器提取卷积层每部分的最大值；(2)average pooling，用过滤器提取卷积层每部分的平均值。一般情况使用max pooling，f=2，s=2，p=0，没有参数需要学习。
* 卷积会缩小尺寸，增加深度；池化只会缩小尺寸。
* example：

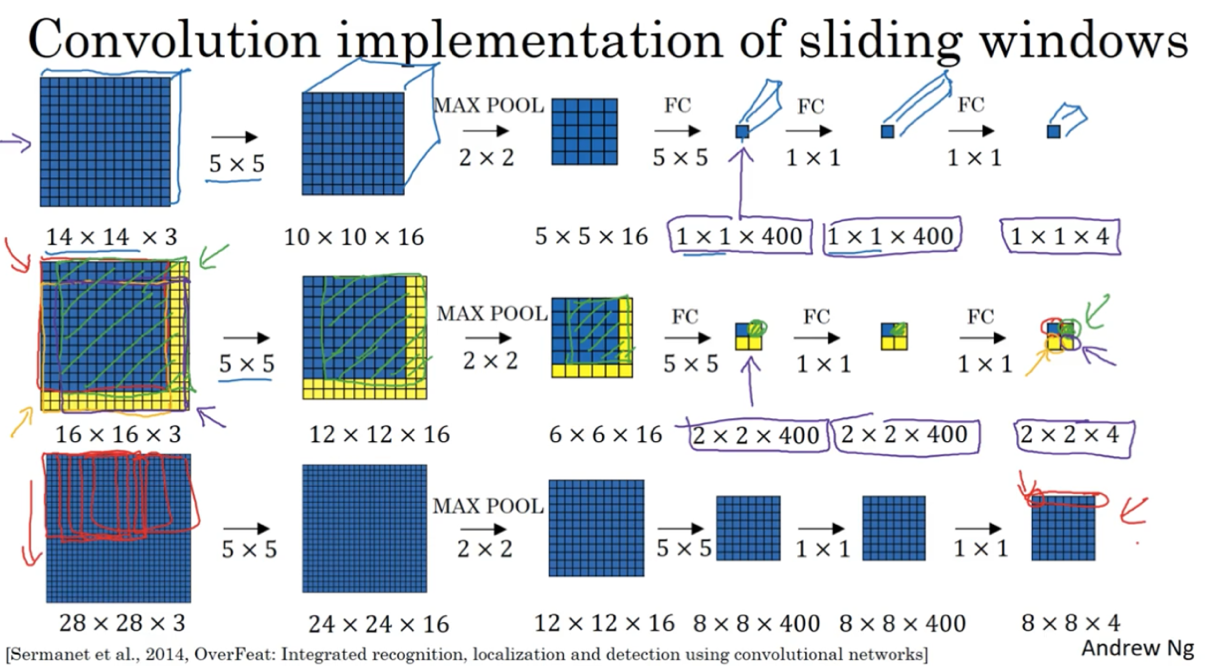


* 反卷积：input=(n\*n)，filter=f，stride=s，padding=p，output=s\*(n-1)-2p+f。

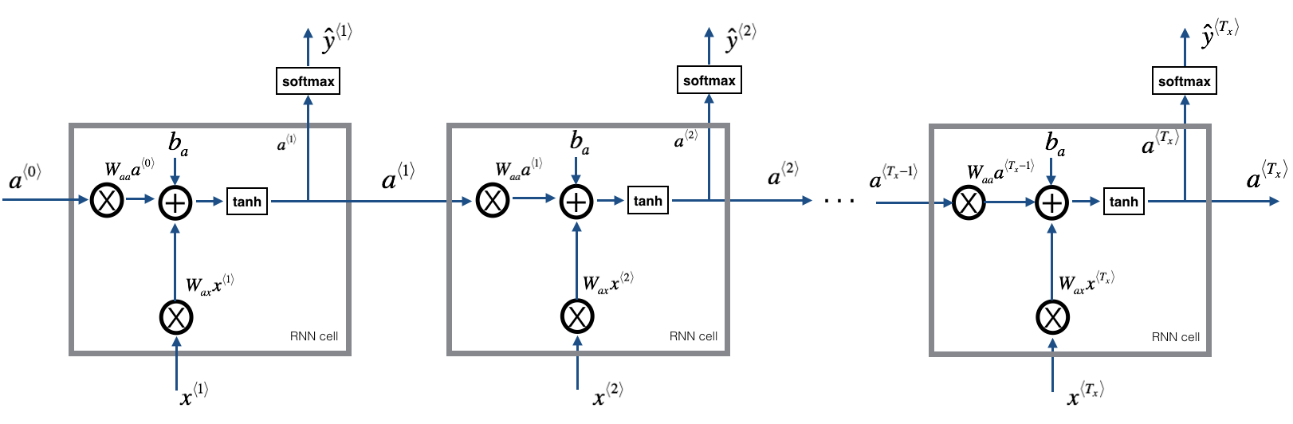


2\*(2-1)-2\*0+3=5

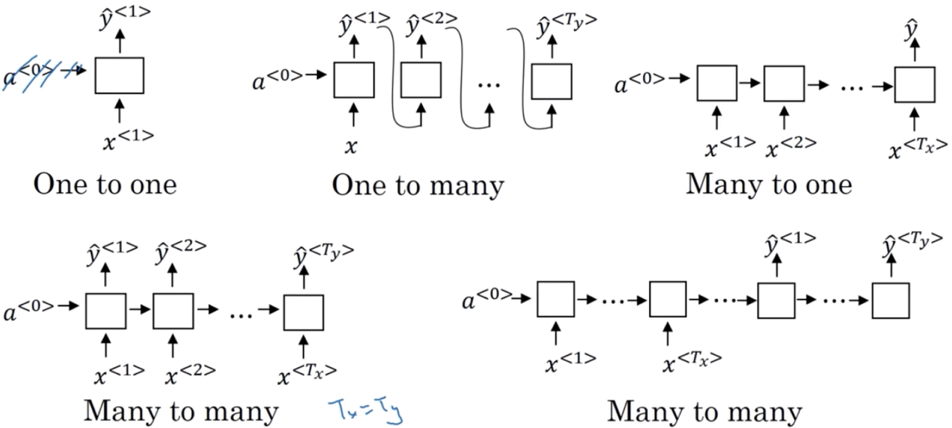
* 1\*1过滤器可以用来减少/增加深度，不改变高度和宽度。
* 数据扩充：可以通过图像垂直翻转、小幅度旋转、截取部分图片、改变颜色等操作扩大数据集。
* 目标检测：训练集需要对带检测目标进行特征标记，之后经过神经网络训练，输出标签需要包涵类别概率和检测目标的位置参数。
* 卷积神经网络滑动窗口实现：



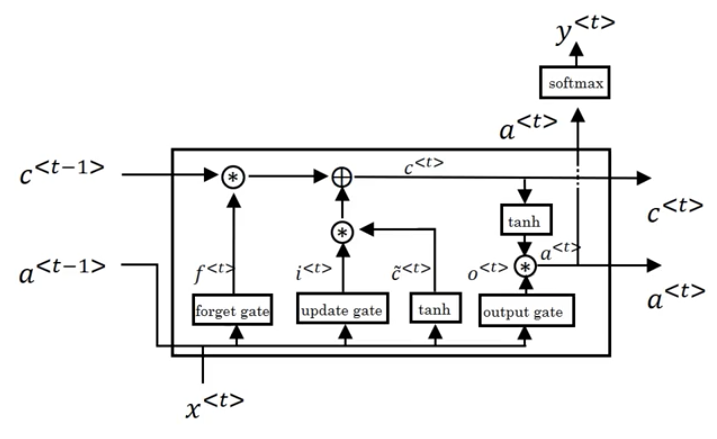
* 交并比IoU：设预测区域为a，实际区域为b，，≥0.5可认为识别准确。
* 非最大值抑制NMS：对于多个预测结果只取概率最大的结果。
* anchor box：用于处理一个窗口内有多个目标，提前制定不同尺寸的anchor box，根据目标比例进行目标检测，一个anchor box则输出h\*w\*c，二个anchor box则输出h\*w\*2c......
* YOLO：通过卷积神经网络进行目标检测，实现分类和回归操作，预测出目标类别与位置，输出[pc,bx,by,bh,bw,c1,c2,c3...]，[0/1，中心x，中心y，高度，宽度，类别1，类别2，类别3]，具体维度取决于anchor box数量。
* 人脸识别：不同人脸图片经过卷积神经网络到全连接层输出等量的若干特征f(x(i))，越小越相似。
* 损失函数；A(anchor)目标人脸，P(positive)相同人脸，N(negative)不同人脸。
* 二分类：假设全连接层f(x(i))有128个特征，。
* 自动编码器：高维->低维->向量->低维->高维，隐藏层具有降维作用，非监督学习模式，不需要label。
* 循环神经网络--------------------------------------------
* 循环神经网络的输入分不同时刻输入，主要解决与时间顺序有关的问题，能够把数据联系在一起。
* 正向传播：，g1=tanh/relu，每个时间步使用相同的waa和wax；，g2=sigmoid/softmax。



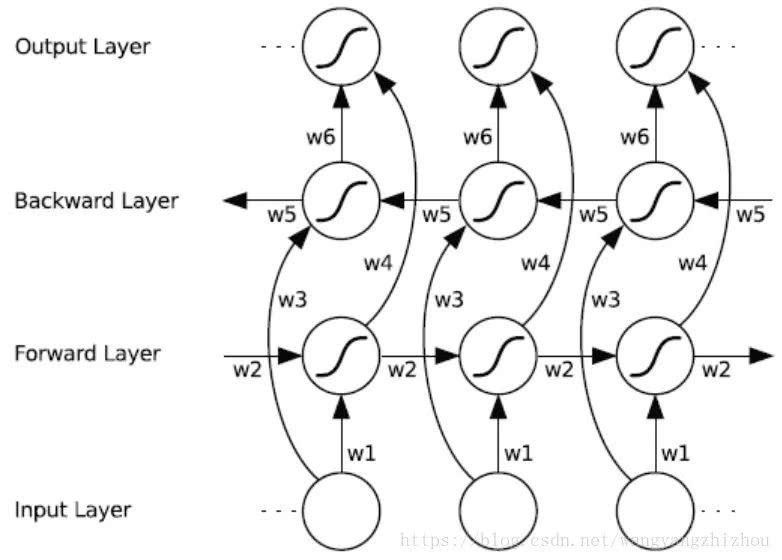
* 使用交叉熵损失函数，将每一个时间步计算出的损失加和获得整个序列的损失。
* 不同类型：



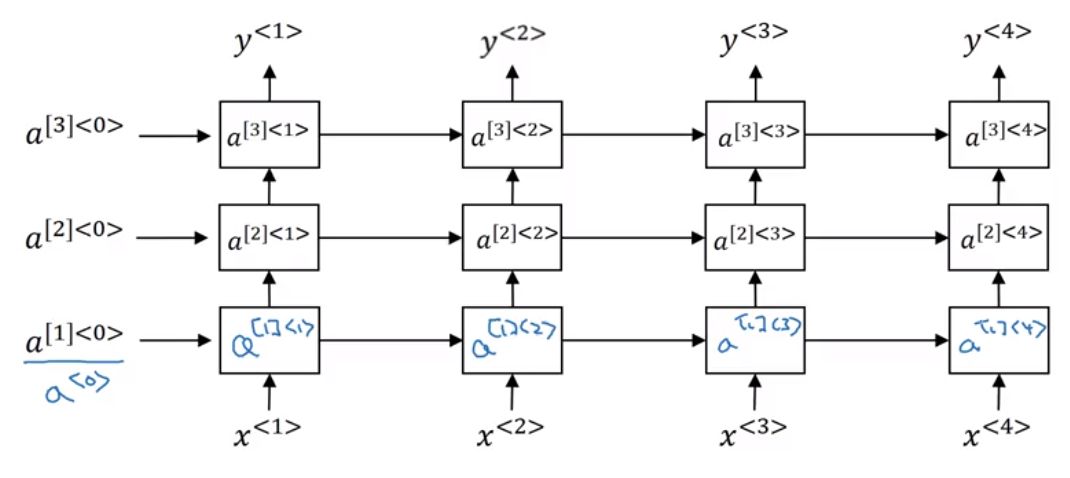
* LSTM：  
  遗忘门：  
  更新门：  
  输出门:  
  更新记忆细胞：  
  输出c(与输出门无关)：  
  输出a：  
  输出y：



* 双向循环神经网络(BRNN)用于联系“过去”与“未来”。



* 深度循环神经网络(DRNN)用于拟合更复杂的数据。



* outputs与state：  
  inputs[batch\_size, max\_taime, input\_size]  
  outputs[batch\_size, max\_time, cell\_size]  
  state[batch\_size, cell\_size] state[0]=c(t) state[1]=h(t)

