深度学习

2019年3月15日　18:12

查准率（precision）和查全率

F_1分数的定义是这个公式：$\frac{2}{\frac{1}{P}+\frac{1}{R}}$

端到端的深度学习
end-to-end deep learning

CUDA_VISIBLE_DEEVICES=1 python xxx.py
指定GPU进行CUDA加速
注意：要大写CUDA_VISIBLE_DEVICES

RMSprop

$$S_{dw}=\beta S_{dw}+(1-\beta)dW^2$$

$$S_{db}=\beta S_{db}+(1-\beta)db^2,$$

$$W:=W-a\frac{dW}{\sqrt{S_{dW}}},\ b:=b-\alpha\frac{db}{\sqrt{S_{db}}},$$

Train/dev/test distributions

让你的开发集和测试集来自同一分布

不是所有程序都可以用CUDA加速的，一般的python程序就不可以，像tf、pt的CDUA加速修改版可以,专门的CUDA编程可以，能快很多。

动量梯度下降法
Momentum

On iteration t:

Compute dW, db on the current mini-batch

$$v_{dW}=\beta v_{dW}+(1-\beta)dW$$

$$v_{db}=\beta v_{db}+(1-\beta)db$$

$$W=W-\alpha v_{dW},\quad b=b-\alpha v_{db}$$

Adam：
Momentum + RMSprop

Adam optimization algorithm

$V_{dw}=0,\ S_{dw}=0.\ V_{db}=0,\ S_{db}=0$

On iteration t:

Compute dW, db using current mini-batch

$V_{dw}=\beta_1 V_{dw}+(1-\beta_1)dW$, $V_{db}=\beta_1 V_{db}+(1-\beta_1)db$ ← "moment" β_1

$S_{dw}=\beta_2 S_{dw}+(1-\beta_2)dW^2$, $S_{db}=\beta_2 S_{db}+(1-\beta_2)db$ ← "RMSprop" β_2

$V^{corrected}_{dw}=V_{dw}/(1-\beta_1^t)$, $V^{corrected}_{db}=V_{db}/(1-\beta_1^t)$

$S^{corrected}_{dw}=S_{dw}/(1-\beta_2^t)$, $S^{corrected}_{db}=S_{db}/(1-\beta_2^t)$

$W:=W-\alpha\frac{V^{corrected}_{dw}}{\sqrt{S^{corrected}_{dw}}+\varepsilon}$　$b:=b-\alpha\frac{V^{corrected}_{db}}{\sqrt{S^{corrected}_{db}}+\varepsilon}$

使用 **Adam** 算法，首先你要初始化，$v_{dW}=0$，$S_{dW}=0$，$v_{db}=0$，$S_{db}=0$，在第t次迭

Hyperparameters choice:

→ α : needs to be tune
→ β_1 : 0.9 → (dw)
→ β_2 : 0.999 → (dw^2)
→ ε : 10^{-8}

Adam: Adaptive moment estimation

学习率衰减

VGG-16

残差网络(ResNets)： 跳跃连接

Batch Norm是将z归一化

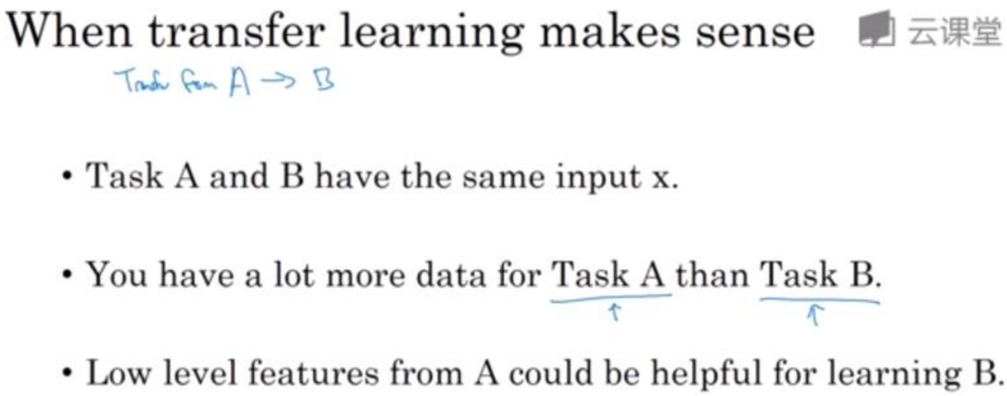
$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$X \xrightarrow{w^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[\text{Batch Norm (BN)}]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \xrightarrow{w^{[2]}, b^{[2]}} z^{[2]} \xrightarrow[\text{BN}]{\beta^{[2]}, \gamma^{[2]}} \tilde{z}^{[2]} \rightarrow a^{[2]} \rightarrow$$

迁移学习（transfer learning）

：1.预训练

2.微调



多任务学习

端到端学习

卷积运算：边缘检测

Padding（图像四周填充）

Same 卷积，那意味你填充后，你的输出大小和输入大小是一样的

Valid 卷积意味着不填充

卷积步长 （stride）：卷积框移动的长度

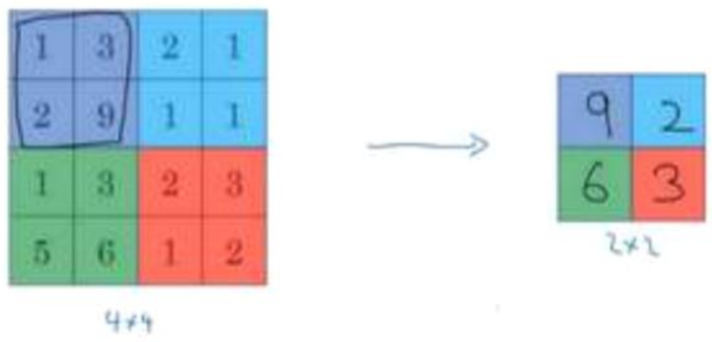
如果你用一个$f \times f$的过滤器卷积一个$n \times n$的图像，

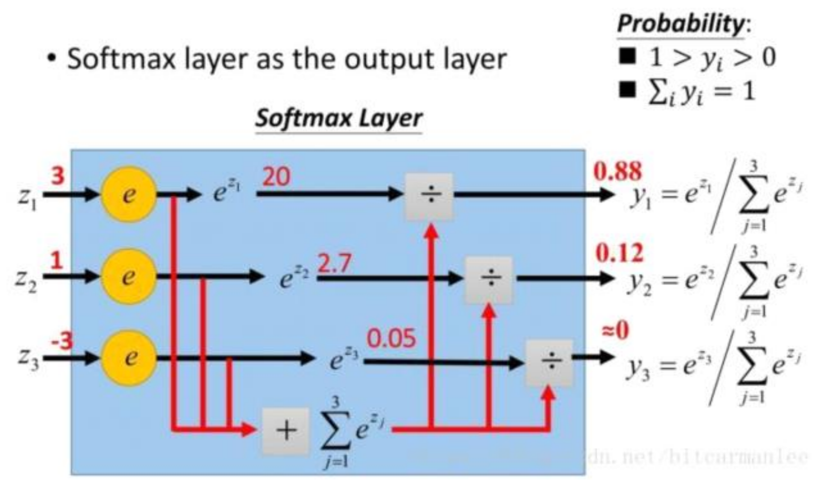
你的 **padding** 为p，步幅为s，

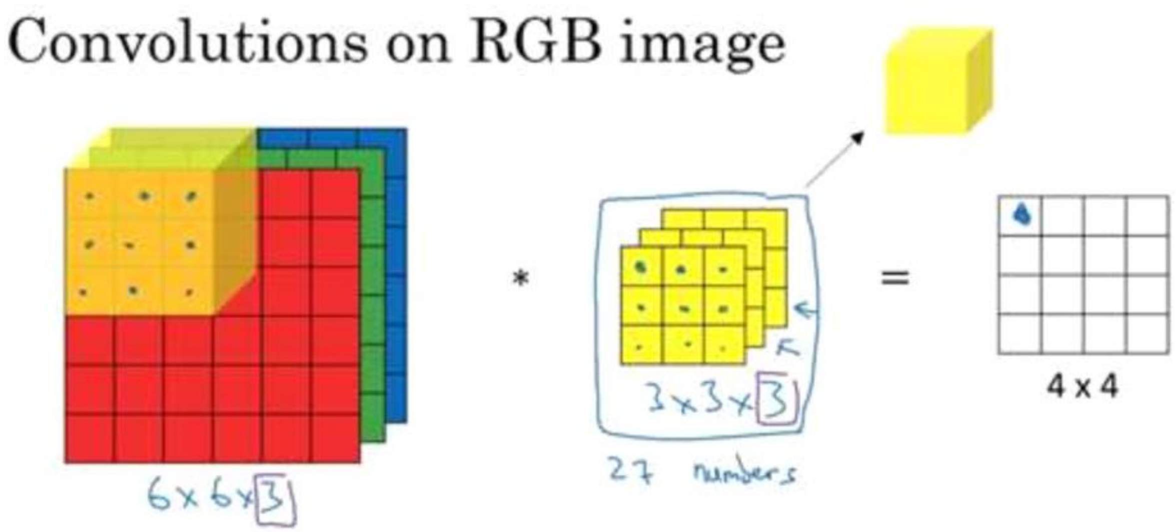
输出于是变为$\frac{n+2p-f}{s} + 1$

池化层（Pooling layers）

Max pooling
超级参数 f=2 s=2







Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

Output: $n_H^{[l]} \times n_W^{[l]} \times \boxed{n_c^{[l]}}$

$$n_{H,W}^{[l]} = \left\lfloor \frac{n_{H,W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

Outline

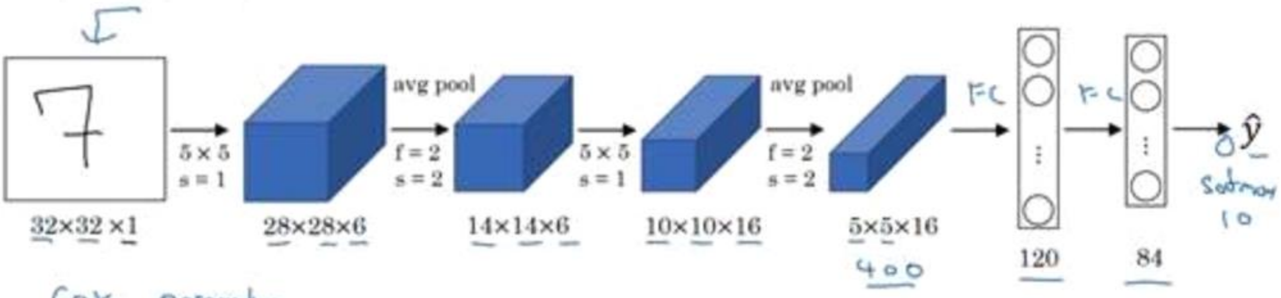
Classic networks:

- LeNet-5
- AlexNet
- VGG

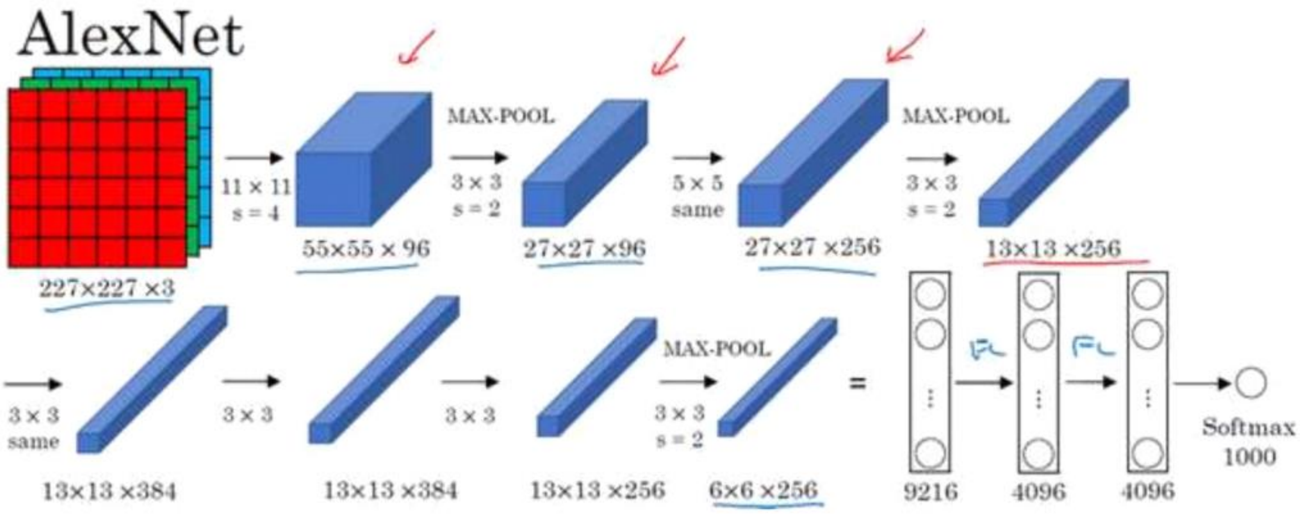
ResNet (152)

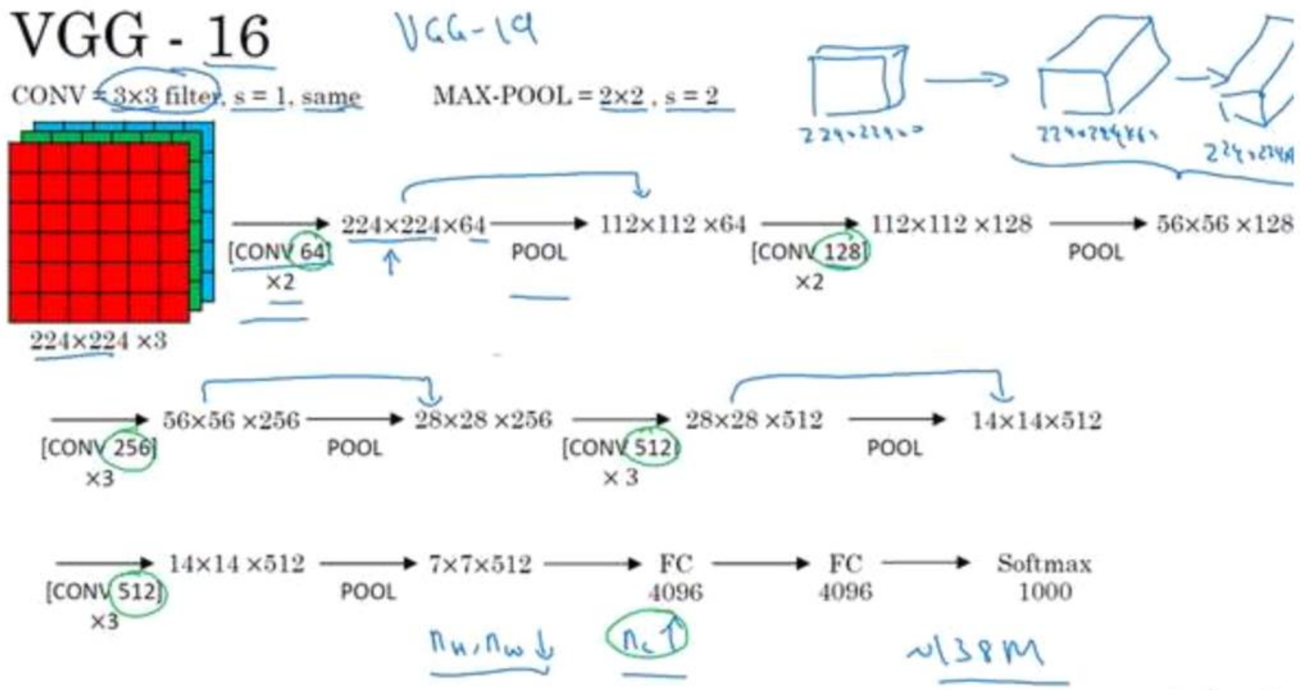
Inception

LeNet - 5

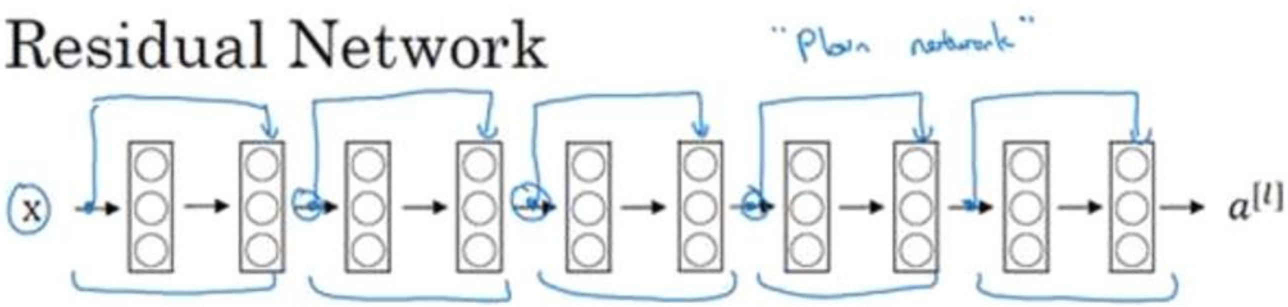


60K parameters

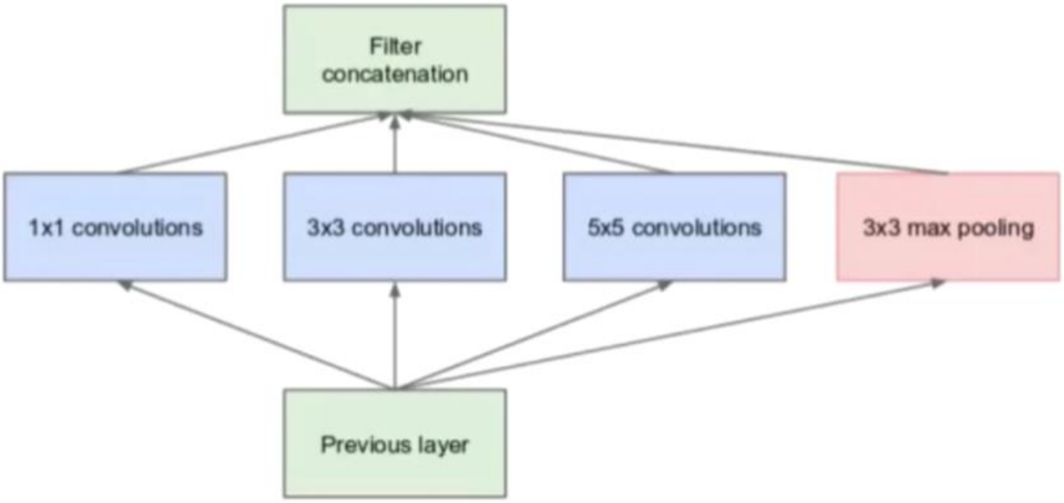




残差网络(ResNets)：跳跃连接（Skip connection）



Inception 网络



(a) Inception module, naïve version

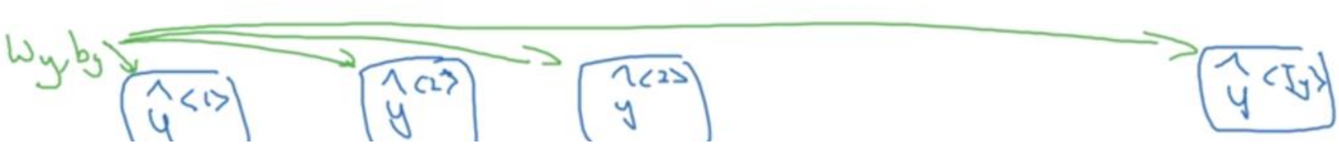
googLeNet是由Inception模块堆积而成:

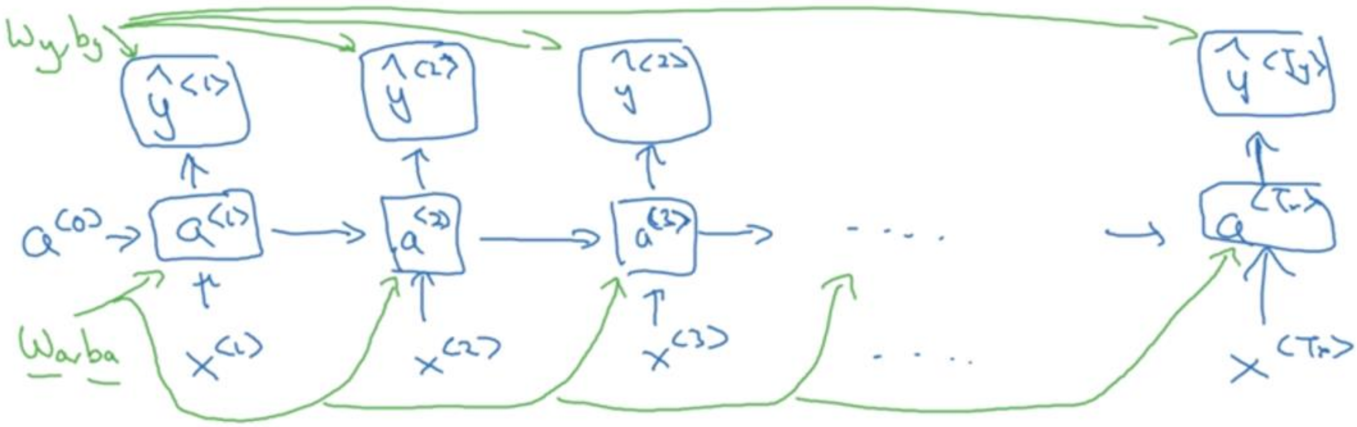
使 用 开 源 的 实 现 方 案

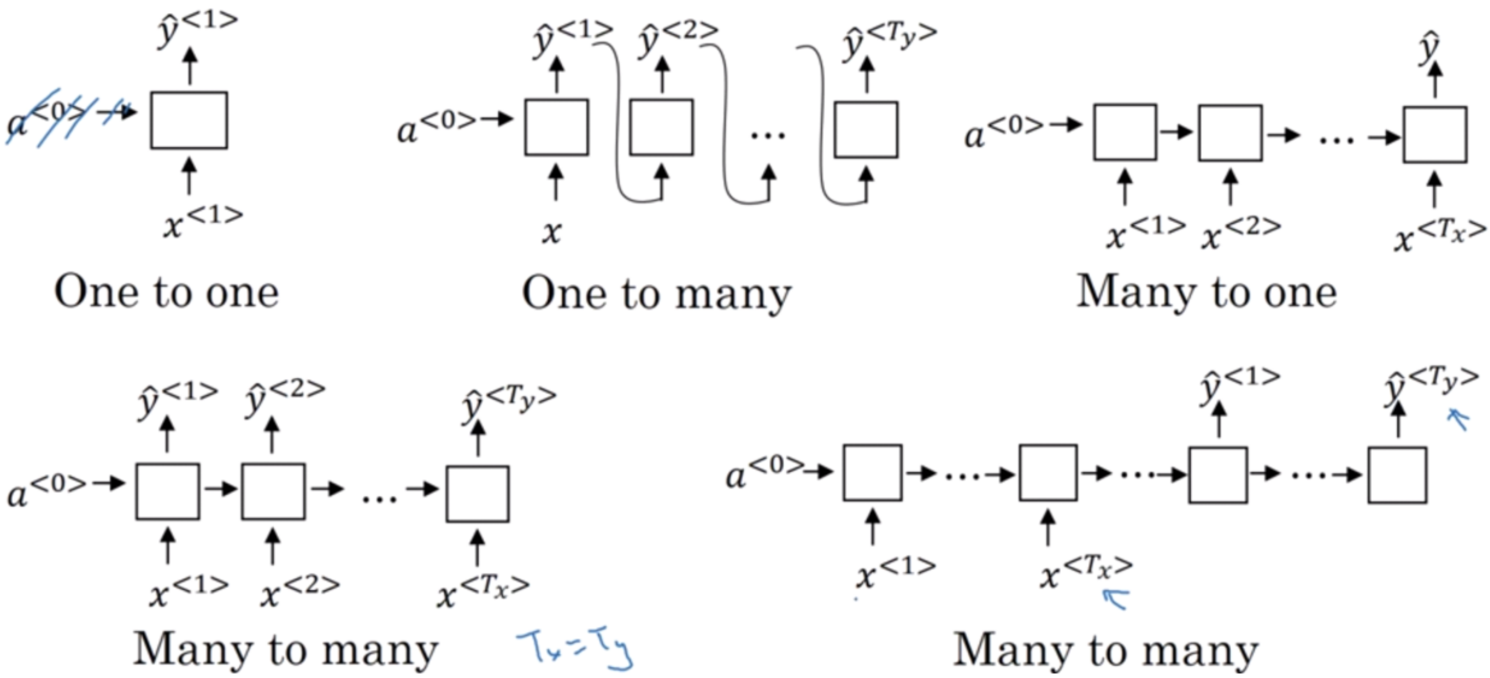
迁移学习:
如果你有越来越多的数据，你需要冻结的层数越少，你能够训练的层数就越多。

序列模型(Sequence Models)

RNN




Wy,by
Wa,ba
a<0>
a<1>
a<2>
a<3>
y<1>
y<2>
y<3>
y<Ty>
x<1>
x<2>
x<3>
x<Tx>


One to one
One to many
Many to one
Many to many
Tx=Ty
Many to many

GRU and LSTM

1 GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \underbrace{(1 - \Gamma_u)}_{\Gamma_f} * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

2 LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

(update) $\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$

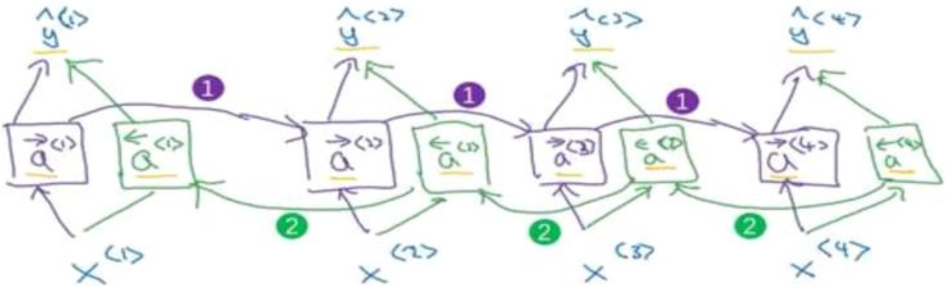
(forget) $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$

(output) $\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

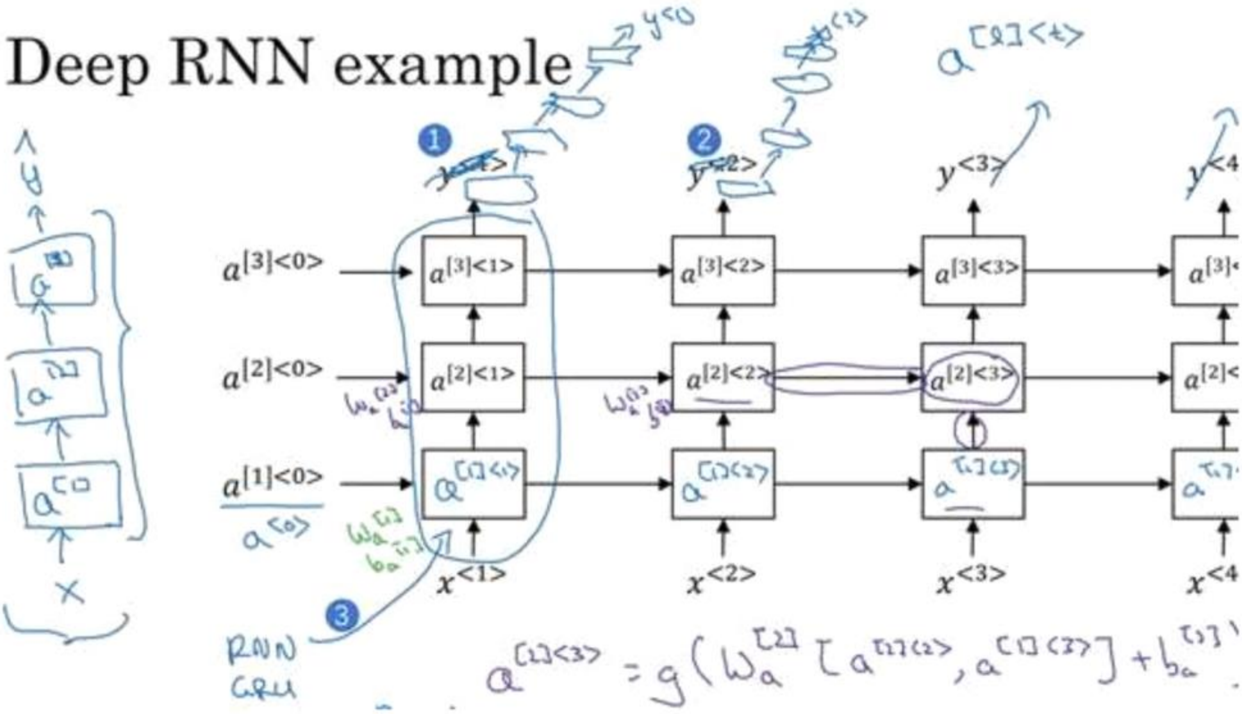
$$a^{<t>} = \Gamma_o * c^{<t>}$$

Bidirectional RNN (BRNN)



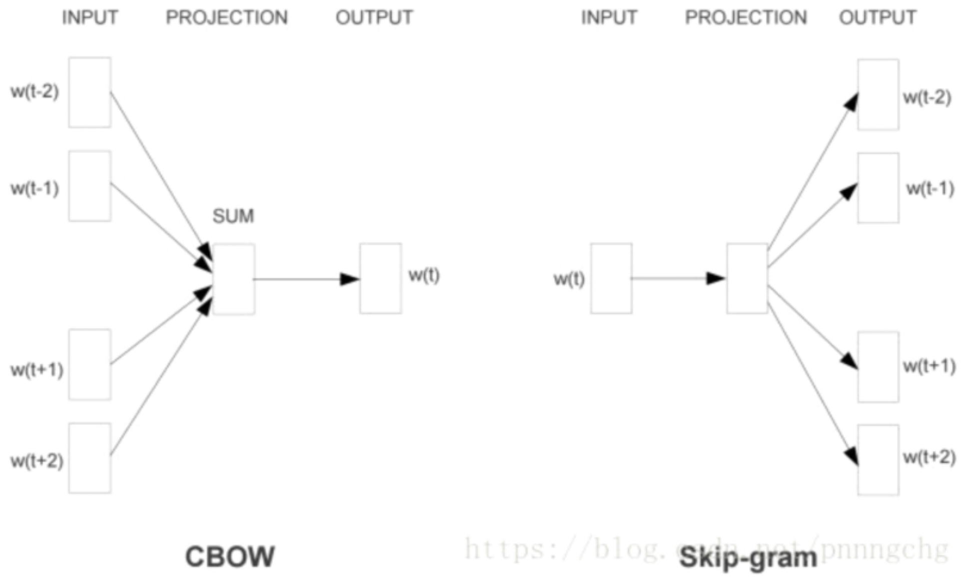
Acyclic graph

Deep RNN example



$$a^{[2]<3>} = g(W_a^{[2]}[a^{[2]<2>}, a^{[1]<3>}] + b_a^{[2]})$$

Word2Vec


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram