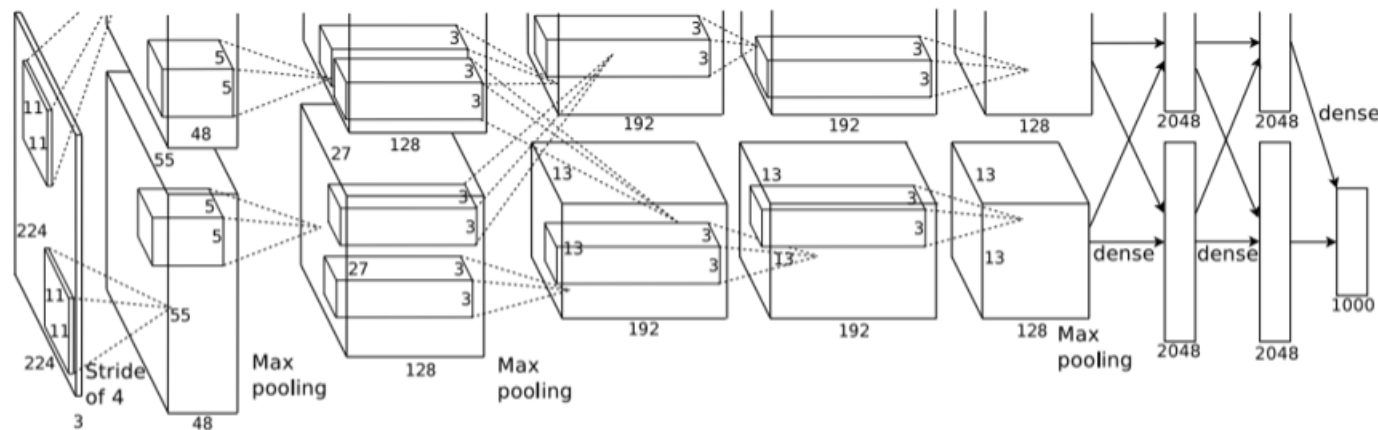


第一部分

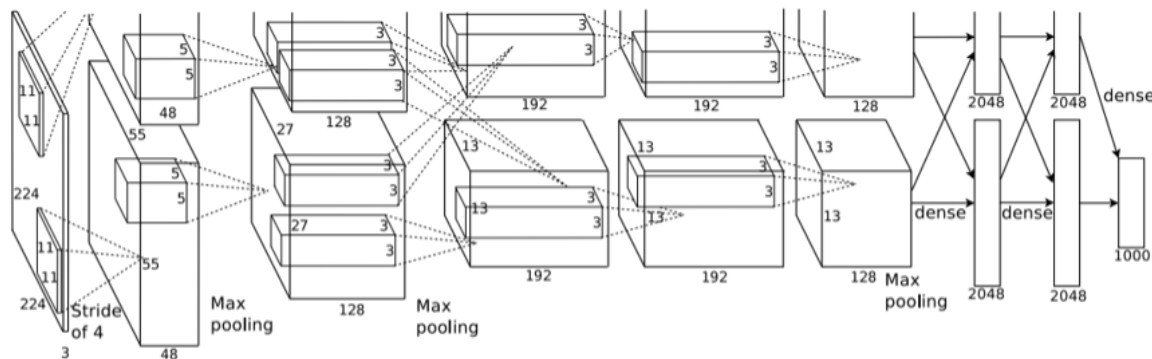
CNN几类常见网络结构

AlexNet



首先这幅图分为上下两个部分的网络，论文中提到这两部分网络是分别对应两个GPU，只有到了特定的网络层后才需要两块GPU进行交互，这种设置完全是利用两块GPU来提高运算的效率。网络总共的层数为8层，5层卷积，3层全连接层。最后一层全连接层的输出是1000维softmax的输入，softmax会产生1000类标签的分布网络包含8个带权重的层

AlexNet



<https://www.cnblogs.com/wangguchangqing/p/10333370.html>

卷积层C1

该层的处理流程是：卷积-->ReLU-->池化-->归一化。

卷积层C2

该层的处理流程是：卷积-->ReLU-->池化-->归一化

卷积层C3

该层的处理流程是：卷积-->ReLU

卷积层C4

该层的处理流程是：卷积-->ReLU 该层和C3类似

卷积层C5

该层处理流程为：卷积-->ReLU-->池化

全连接层FC6

该层的流程为：（卷积）全连接 -->ReLU --->Dropout

全连接层FC7

流程为：全连接-->ReLU-->Dropout

输出层

VGG

<https://www.jianshu.com/p/15e413985f25>

VGGNet使用的全部都是3x3的小卷积核和2x2的池化核，通过不断加深网络来提升性能。各个级别VGG的模型结构如表所示，其下方为不同模型的参数数量。可以看到，虽然从A到E每一级网络逐渐变深，但是网络的参数量并没有增长很多，这是因为参数量主要都消耗在最后3个全连接层了。前面的卷积层数量虽多，但是参数量其实都不大，不过训练时候耗时的依然是卷积层，因为这部分计算量比较大。其中D, E分别为VGG16和VGG19。

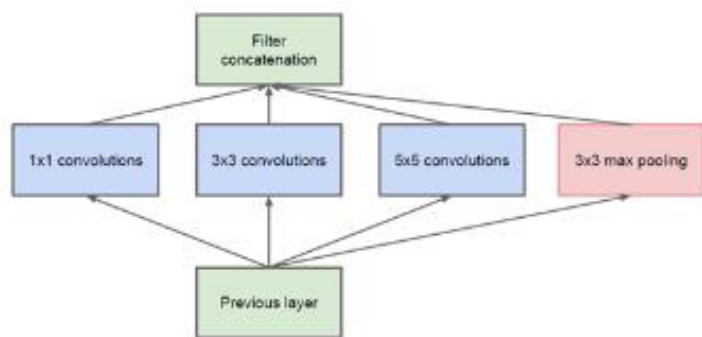
ConvNet Conguration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 L R N	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

https://www.csdn.net/wsp_1138880114

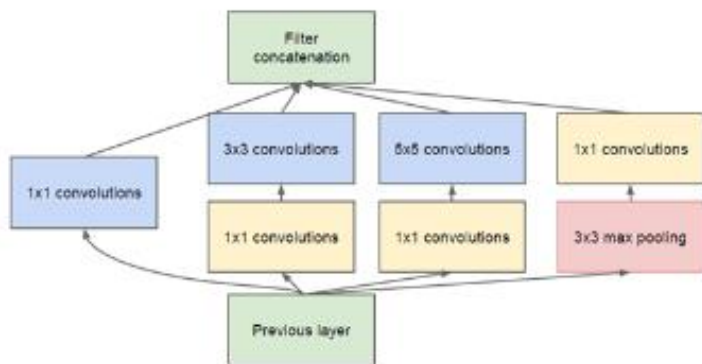
GoogLeNet网络

<https://my.oschina.net/u/876354/blog/1637819>

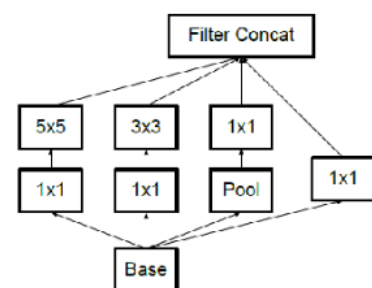
- Inception历经了V1、V2、V3、V4等多个版本的发展，不断趋于完善



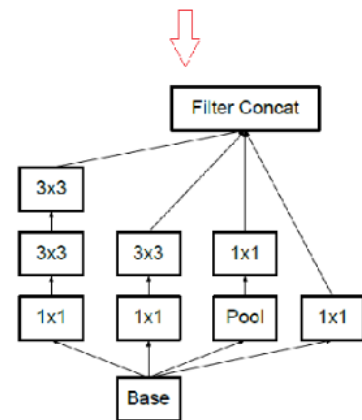
Inception V1



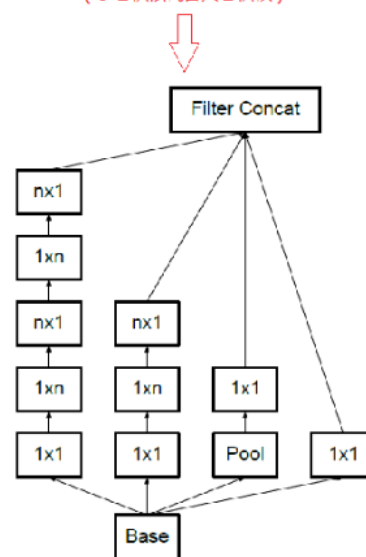
Inception V2



Inception V1 结构



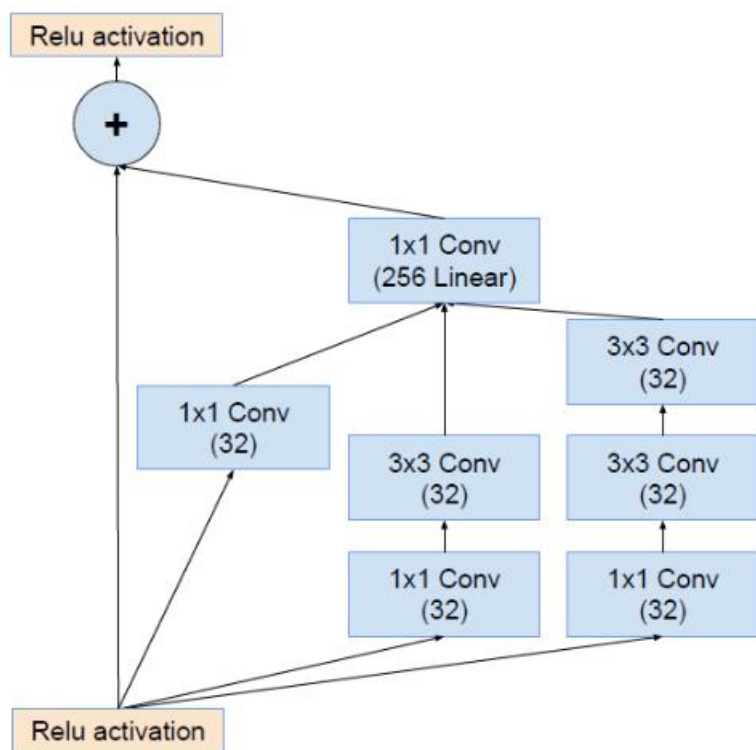
用 3x3 卷积核代替 5x5 卷积核
(小卷积核代替大卷积核)



用 1xn 和 nx1 卷积核代替 nxn 卷积核

Inception V2 结构

大尺度的卷积往往会造成计算的浪费，因为大尺度卷积可以分解为几个小尺度的卷积，从而减小计算量。例如5x5的卷积可以分解为两层3x3的卷积，而后者的计算量也更小。因此，在inception v2中，大尺度的卷积被分解为小尺度卷积。此外，论文还提出了使用1xn和nx1的两层卷积代替nxn卷积。



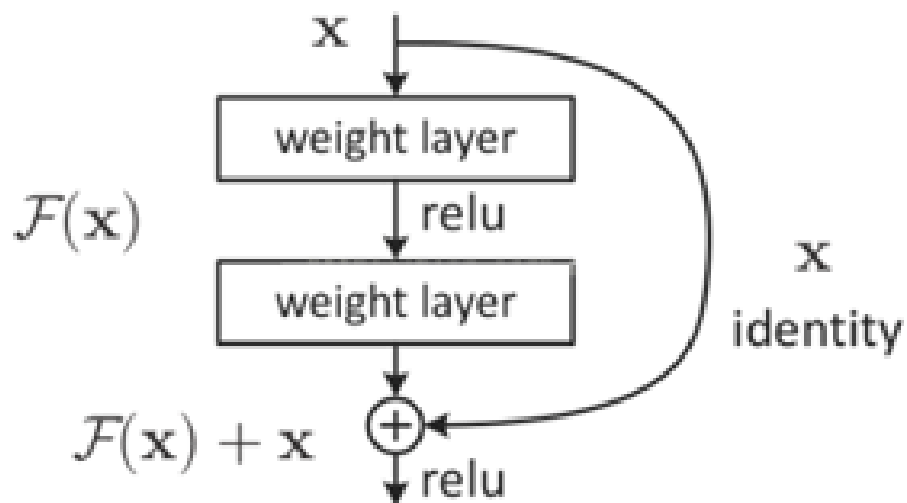
Inception V4

- Inception V4研究了Inception模块与残差连接的结合。ResNet结构大大地加深了网络深度，还极大地提升了训练速度，同时性能也有提升
- Inception V4主要利用残差连接（Residual Connection）来改进V3结构

Resnet

<https://my.oschina.net/u/876354/blog/1622896>

残差块的结构



在上图的残差网络结构图中，通过“**shortcut connections (捷径连接)**”的方式，直接把输入 x 传到输出作为初始结果，输出结果为 $H(x) = \mathcal{F}(x) + x$ ，当 $\mathcal{F}(x) = 0$ 时，那么 $H(x) = x$ ，也就是上面所提到的恒等映射。于是，ResNet相当于将学习目标改变了，不再是学习一个完整的输出，而是目标值 $H(x)$ 和 x 的差值，也就是所谓的**残差 $\mathcal{F}(x) := H(x) - x$** ，因此，后面的训练目标就是要将残差结果逼近于0，使到随着网络加深，准确率不下降。

第二部分

结构特点与优缺点

AlexNet网络特点

<https://baike.baidu.com/item/AlexNet/22689612?fr=aladdin>

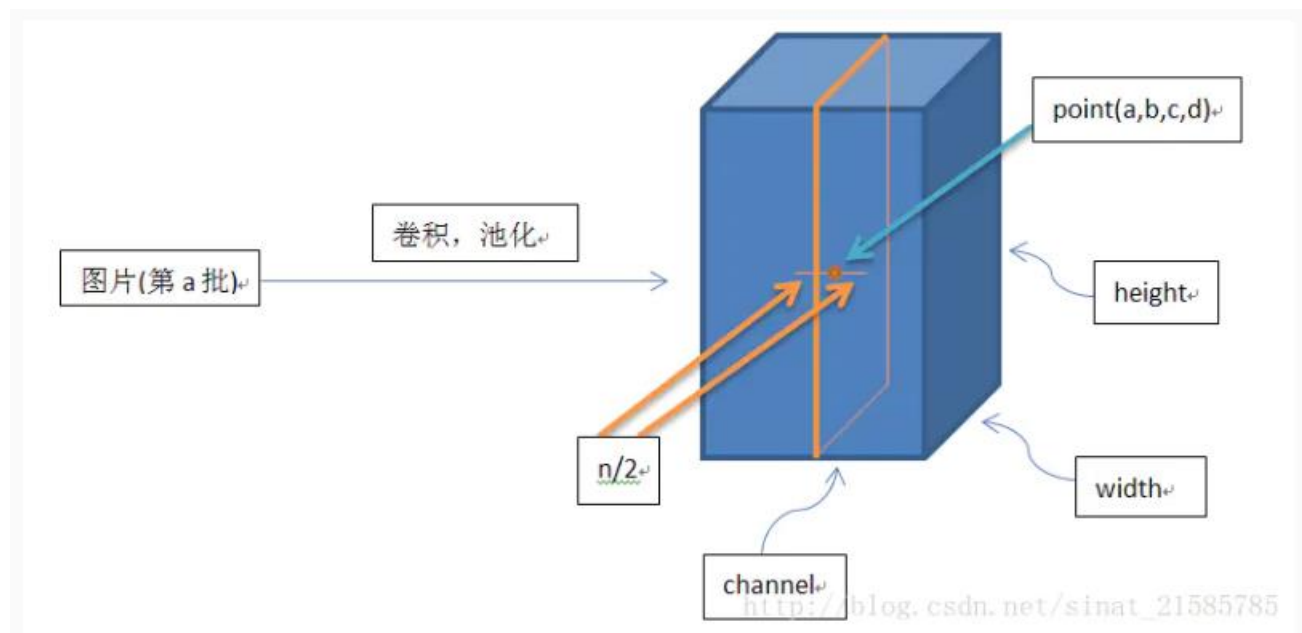
- ①. 使用Relu作为CNN的激活函数。解决了Sigmoid在网络较深时梯度弥散的问题，加快了训练速度。
- ②. 使用 Dropout 随机忽略部分神经元。AlexNet 将其实用化，证实了它的效果。AlexNet 中最后几个全连接层使用了 Dropout，避免模型的过拟合。
- ③. 使用重叠的最大池化操作。AlexNet 全部使用最大池化，避免了平均池化的模糊化效果。AlexNet 提出了让步幅比池化核的尺寸小，这样池化层的输出之间会有重叠和覆盖，提升了特征的丰富性。
- ④. 提出局部响应归一化 LRN (Local Response Normalization) 层。对局部神经元的活动创建竞争机制，使得其中响应比较大的值变得相对更大，并抑制其反馈较小的神经元，增强了模型的泛化能力。有助于快速收敛。
- ⑤. 数据增强。随机地从256*256的原始图像中截取224*224大小的区域（以及水平翻转的镜像），相当于增加了 $2 \times (256 - 224)^2 = 2048$ 倍的数据量。预测时，取图像的四个角加中间共 5 个位置，并进行左右翻转，一共获得 10 张图像，对它们进行预测并对 10 次结构求均值。

LRN层

一般是在激活、池化后进行的一种处理方法。

<https://www.jianshu.com/p/c014f81242e7>

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$



举个例子:

i=10, N=96, n=4。当求第i=10个卷积核在位置x,y处提取到的特征(a), 局部响应归一化过程如下: 用(a)除以 第8/9/10/11/12在位置x,y处提取的特征之和。

使用LRN层的好处?

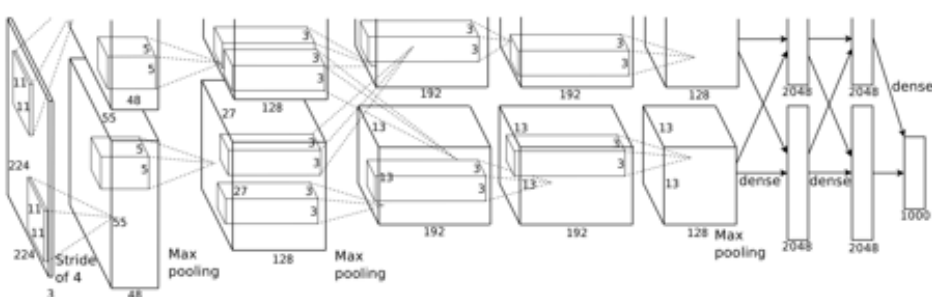
<https://blog.csdn.net/cc1949/article/details/79238405>

首先要理解归一化的好处。归一化可以加快收敛速度。

而LRN即对一个输入的局部区域进行归一化。

在训练大量的训练数据过程中, 一旦每批训练数据的分布各不相同 (batch 梯度下降), 那么网络就要求在每次迭代都去学习适应不同的分布, 这样将会大大降低网络的训练速度, 这也正是为什么我们需要对数据都要做一个归一化预处理的原因。

3.3 LRN(Local Response Normalization) 局部响应值归一化

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$


局部归一的动机：在神经生物学有一个概念叫做 侧抑制（lateral inhibito），指的是 被激活的神经元抑制相邻神经元。归一化（normalization）的目的是“抑制”，局部响应归一化就是借鉴 侧抑制 的思想来实现局部抑制，尤其当我们使用ReLU 的时候这种“侧抑制”很管用。

好处：有利用增加范化能力，做了平滑处理，识别率提高1~2%

LRN层模仿生物神经系统的侧抑制机制，对局部神经元的活动创建竞争机制，使得响应比较大的值相对更大，提高模型范化能力。Hinton在Imagenet中表明分别提升1.4%和1.2%。

a表示第i个核在位置（x,y）运用ReLU非线性化神经元输出，n是同一位置上临近的kernel m ap的数目，N是kernal的总数。

K=2,n=5,alpha=1*e-4, beta=0.75

1.[What Is Local Response Normalization In Convolutional Neural Networks?](http://blog.csdn.net/hduxiejun)

VGG网络

特点：

VGG16相比AlexNet的一个改进是采用连续的几个3x3的卷积核代替AlexNet中的较大卷积核（11x11，7x7，5x5）。对于给定的感受野（与输出有关的输入图片的局部大小），采用堆积的小卷积核是优于采用大的卷积核，因为多层非线性层可以增加网络深度来保证学习更复杂的模式，而且代价还比较小（参数更少）。

优点：

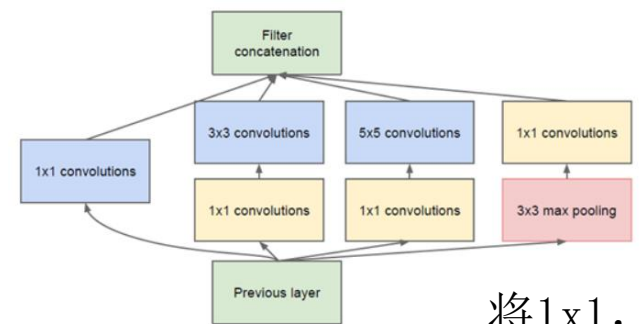
- ①. VGGNet的结构非常简洁，整个网络都使用了同样大小的卷积核尺寸（3x3）和最大池化尺寸（2x2）。
- ②. 几个小滤波器（3x3）卷积层组合比大滤波器（5x5或7x7）卷积层好。
- ③. 验证了通过不断加深网络结构可以提升性能。

缺点：

- ①. VGG耗费更多计算资源，并且使用了更多的参数，导致更多的内存占用（140M）。其中绝大多数的参数都是来自于第一个全连接层。

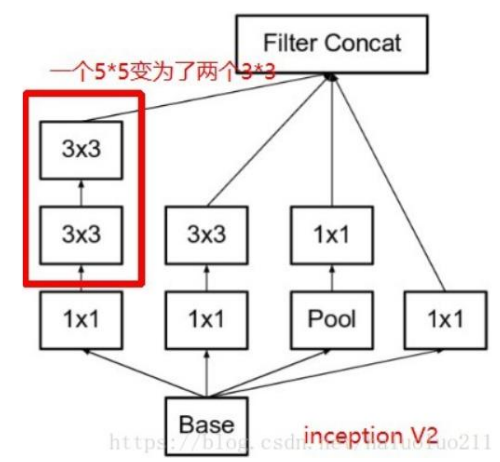
GoogLeNet网络

<https://blog.csdn.net/haluoluo211/article/details/81710799>



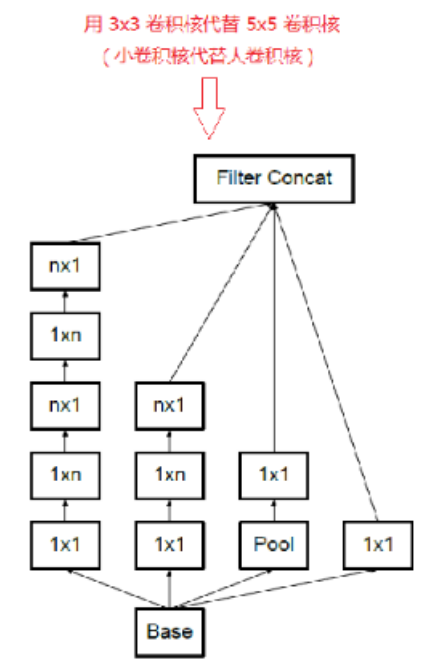
InceptionV1

将1x1, 3x3, 5x5的conv和3x3的pooling, 堆叠在一起, 一方面增加了网络的width, 另一方面增加了网络对尺度的适应性.



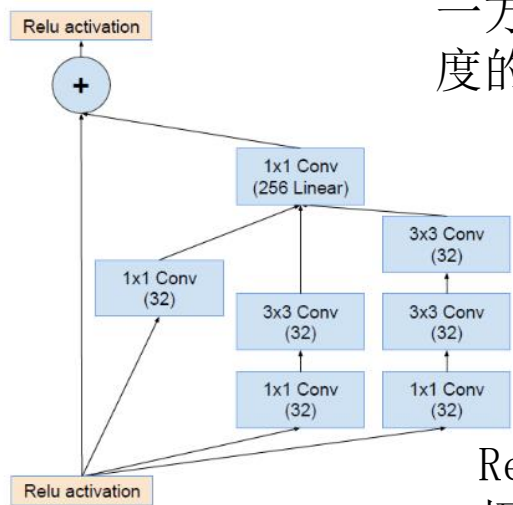
<https://blog.csdn.net/haluoluo211>

InceptionV2/V3



用 1xn 和 nx1 卷积核代替 nxn 卷积核

Inception V2 结构



InceptionV4

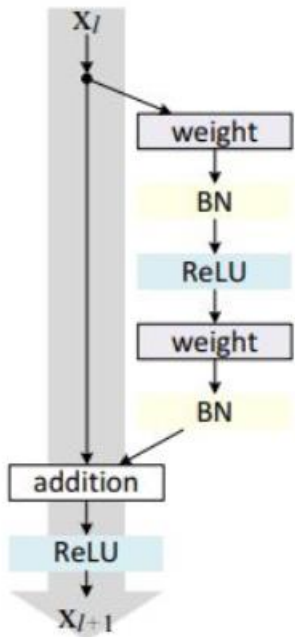
ResNet的结构可以极大地加速训练, 同时性能也有提升

参数更少: GoogleNet参数为500万个, AlexNet参数个数是GoogleNet的12倍, VGGNet参数又是AlexNet的3倍;
性能更好: 占用更少的内存和计算资源, 且模型结果的性能却更加优越。

ResNet网络 <https://zhuanlan.zhihu.com/p/42706477>

残差块=直接映射部分+残差部分

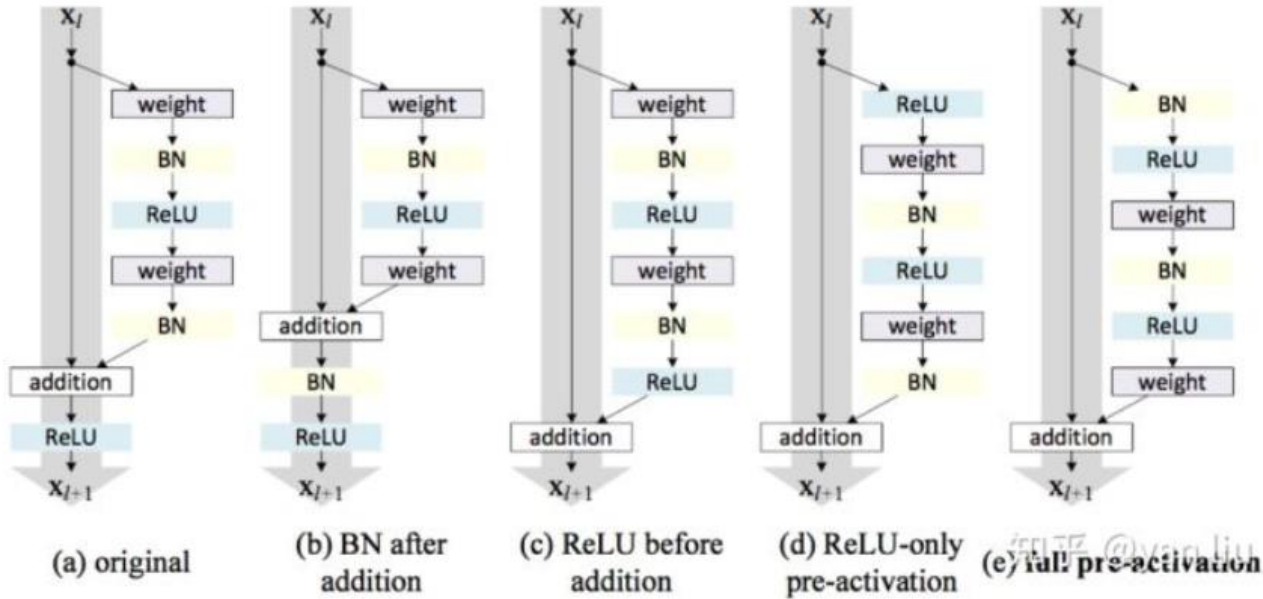
$$x_{l+1} = x_l + \mathcal{F}(x_l, W_l)$$



直接映射是最好的选择

激活函数的位置

实验结果表明将激活函数移动到残差部分可以提高模型的精度。即下图 (e)。

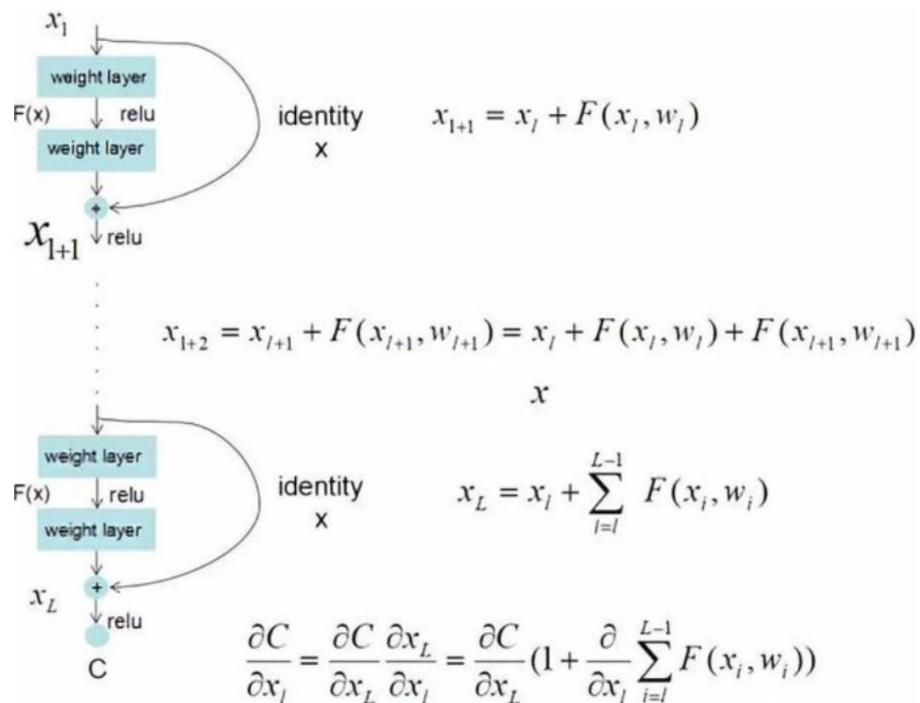


特点:

ResNet的直接映射的加入，保证了深层的网络一定比浅层包含更多的图像信息。残差网络的特点是容易优化，并且能够通过增加相当的深度来提高准确率。其内部的残差块使用了跳跃连接，解决了深度网络的退化问题和梯度消失问题。

ResNet网络 <https://www.cnblogs.com/gczr/p/10127723.html>

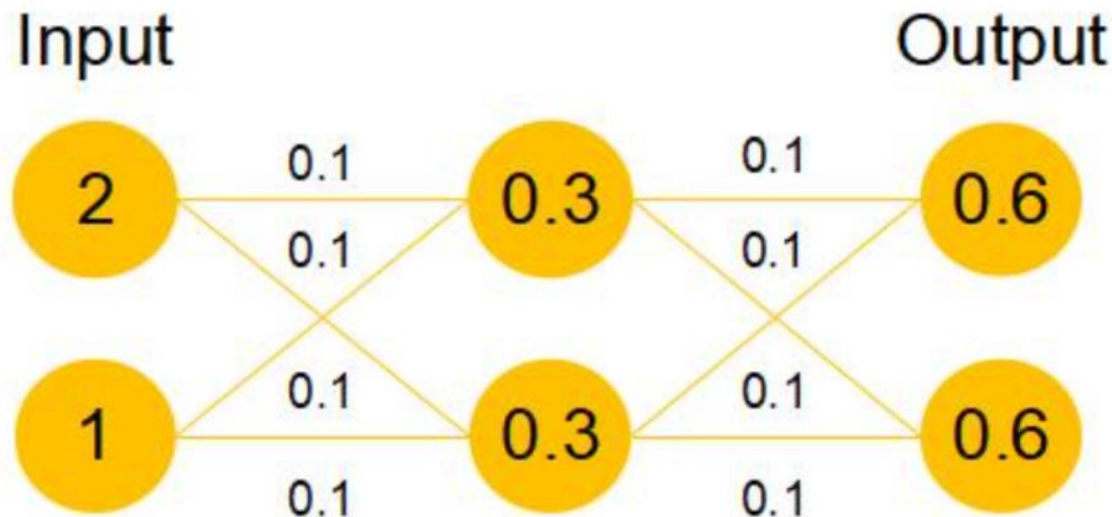
1.为什么可以解决梯度消失问题？



$\frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} F(x_i, w_i)$ 在整个训练过程中不可能一直为-1，也就是说在残差网络中不会出现梯度消失的问题。

$\frac{\partial \epsilon}{\partial x_L}$ 表示L层的梯度可以直接传递到任何一个比它浅的层。

2.为什么可以解决网络退化问题？



假设该层是冗余的，
引入ResNet之前，需要让该层学习到的参数满足 $h(x)=x$ 。
而ResNet中 $h(x)=F(x)+x$ ；只需要学习参数使得 $F(x)=0$ 。

因为一般每层网络中的参数初始化偏向于0，这样在相比于更新该网络层的参数来学习 $h(x)=x$ ，该冗余层学习 $F(x)=0$ 的更新参数能够更快收敛。

第三部分

损失函数、激活函数、评估指标

深度学习中损失函数是整个网络模型的“指挥棒”，**通过对预测样本和真实样本标记产生的误差反向传播指导网络参数学习。**

激活函数的主要作用是**提供网络的非线性建模能力，分层的非线性映射学习能力。**几乎所有的连续可导函数都可以用作激活函数，但目前常见的多是**分段线性和具有指数形状的非线性函数。**

一个深度学习模型在各类任务中的表现都需要定量的指标进行评估，才能够进行横向的PK比较。

损失函数介绍（分类任务）

假设某分类任务共有 N 个训练样本，针对网络最后分层第 i 个样本的输入特征为 \mathbf{x}_i ，其对应的标记为 Y_i 是最终的分类结果（ C 个分类结果中的一个）， $\mathbf{h} = (h_1, h_2, \dots, h_C)$ 为网络的最终输出，即样本 i 的预测结果。其中 C 是最后所有分类的数量。

1.交叉熵损失函数：通过指数化变换使网络输出 \mathbf{h} 转换为概率形式

$$L_{cross\ entropy\ loss} = L_{softmax\ loss} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{h_{y_i}}}{\sum_{j=1}^C e^{h_j}}$$

2.合页损失函数：广泛在支持向量机中使用，有时也会在损失函数中使用

缺点：对错误越大的样本施以更严重的惩罚，会导致损失函数对噪音敏感。如果一个样本的标记错误或者是离群点，则由于错分导致分类误差很大，影响整个分类超平面的学习，从而降低模型泛化能力。

$$L_{hinge\ loss} = \frac{1}{N} \sum_{i=1}^N \max\{0, 1 - h_{y_i}\}$$

损失函数介绍（分类任务）

3.坡道损失函数：针对噪声数据和离群点具备良好的抗噪特性

优点：克服了合页损失函数鲁棒性差的特点，对噪声数据和离群数据有很好的抗噪能力。因此也被称作鲁棒损失函数。这类损失函数的特点是在分类（回归）问题误差较大区域进行了截断，使得较大的误差不再影响整个损失函数。

s 指定了“截断点”的位置

s 取值最好根据分类任务的类别数 C 而定，一般设置为 $s = -1/(C-1)$

$$L_{ramp\ loss} = L_{hinge\ loss} - \frac{1}{N} \sum_{i=1}^N \max\{0, s - h_{yi}\} = \frac{1}{N} \sum_{i=1}^N (\max\{0, 1 - h_{yi}\} - \max\{0, s - h_{yi}\})$$

交叉熵损失函数，合页损失函数和坡道损失函数只是简单衡量模型预测值与样本真实值之间的误差从而指导训练。他们并没有显示的将特征判别性学习考虑到整个网络训练中，对此，为了进一步提高学习到的特征表示的判别性，近期研究者设计了一些新的损失函数

损失函数介绍（分类任务）

4. 大间隔交叉熵损失函数：

传统的softmax中将输出结果 h 表示为全连接层参数 W 与该层特征向量 x_i 的内积

$$L_{softmax\ loss} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{\|W_i\| \|x_i\| \cos(\theta_{y_i})}}{\sum_{j=1}^C e^{\|W_j\| \|x_i\| \cos(\theta_j)}}$$

将第 i 类分类间隔拉大，得到大间隔交叉熵损失函数

$$L_{large_margin\ softmax\ loss} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{\|W_i\| \|x_i\| \phi(\theta_{y_i})}}{e^{\|W_i\| \|x_i\| \phi(\theta_{y_i})} + \sum_{j \neq y_i} e^{\|W_j\| \|x_i\| \cos(\theta_j)}}$$

大间隔交叉熵损失函数扩大了类间的距离，由于它不仅要求分类正确且要求分开的类保持较大的间隔。使得目标比传统交叉熵更加困难。训练目标变得困难带来的一个额外的好处就是起到防止过拟合的作用。并且在分类性能方面，大间隔交叉熵损失函数要优于交叉熵损失函数和合页函数。

损失函数介绍（分类任务）

5.中心损失函数：中心损失函数在考虑类间距离的同时还将一些注意力放在减少类间差异上。

$$L_{center\ loss} = \frac{1}{2} \sum_{i=1}^N \|x_i - c_{y_i}\|_2^2$$

c_{y_i} 为第 y_i 类所有深度特征的均值（中心），因此叫做中心损失函数。

主要考虑控制类内差异，与考虑类间距离的损失函数配合（交叉熵损失函数）

直观上，上式迫使所有隶属于 y_i 类的样本和中心不要太远，否则将增大惩罚。

由于中心损失函数本身考虑类内差异，因此中心损失函数应与主要考虑类间的损失函数配合使用，如交叉熵函数

$$L_{final} = L_{cross\ entropy\ loss} + L_{center\ loss}(h, y_i) = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{h_{y_i}}}{\sum_{j=1}^C e^{h_j}} + \frac{\lambda}{2} \sum_{i=1}^N \|x_i - c_{y_i}\|_2^2$$

式中 λ 为两个损失函数的调节项， λ 越大，类内差异比重越大

损失函数介绍（分类任务）

6.Focal loss: 解决多分类任务中样本不平衡的现象，论文中 $\alpha=0.25$ ， $\gamma=2$ 效果最好。

$$\text{soft max}(y_i) = \frac{e^{y_i}}{\sum_{i=1}^n e^{y_i}}$$

$$L_{\text{focal_loss}}(y, y_hat) = -\frac{1}{n} \sum_{i=1}^n y_hat_i \times \partial_i \times (1 - \text{soft max}(y_i))^\gamma \log(\text{soft max}(y_i))$$

7.dice loss: 2分类任务时使用的loss，本质就是不断学习，使得交比并越来越大。

8. ROC AUC Score: 使用 Wilcoxon-Mann-Whitney Statistic统计量来逼近ROC曲线下面积

(<http://tflearn.org/objectives/#roc-auc-score>)

9. Weak Softmax Crossentropy 2d: 使用弱交叉熵计算图像分割方面的损失

(<http://tflearn.org/objectives/#weak-crossentropy-2d>)

10. Contrastive Loss : 对比损失函数，主要用在Siamese network中

(<http://yann.lecun.com/exdb/publis/pdf/chopra-05.pdf>)

激活函数介绍

1.sigmoid: sigmoid 的软饱和性，使深度神经网络难以有效训练,易出现梯度消失

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \text{sigmoid}'(x) = \text{sigmoid}(x)(1 - \text{sigmoid}(x))$$

2.tanh: 收敛速度比sigmoid快；可降低迭代次数

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} = 2\text{sigmoid}(2x) - 1$$

3.ReLU: 可缓解梯度消失；存在偏移现象和神经元死亡，会共同影响网络的收敛性

$$\text{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

4.PReLU: 虽然引入额外参数，但基本不需担心过拟合；与ReLU相比，PReLU收敛速度更快

$$\text{PReLU}(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$$

激活函数介绍

5.Maxout: 够缓解梯度消失;规避了ReLU神经元死亡的缺点;但增加了参数和计算量

$$Maxout(x) = \max(w_1x + b_1, w_2x + b_2, \dots, w_nx + b_n)$$

6.ELU:右侧线性能缓解梯度消失,左侧软饱和能对输入变化或噪声更鲁棒。ELU的输出均值接近于零,所以收敛速度更快。

$$ELU(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$$

7.Softplus

$$f(x) = \log(e^x + 1)$$

8.Softsign

$$f(x) = \frac{x}{|x| + 1}$$

总结:一般在分类问题上,建议先尝试ReLU,其次ELU,这是两类不引入额外参数的激活函数。然后考虑用具备学习能力的PReLU,并使用正则化技术,例如应该考虑在网络中增加Batch Normalization层。

评估指标（二分问题）

计标签为正样本，分类为正样本的数目为**True Positive**，简称**TP**。

标签为正样本，分类为负样本的数目为**False Negative**，简称**FN**。

标签为负样本，分类为正样本的数目为**False Positive**，简称**FP**。

标签为负样本，分类为负样本的数目为**True Negative**，简称**TN**。

判别是否为正例需设概率阈值**T**，预测概率大于阈值**T**的为正类，小于阈值**T**的为负类，默认为**0.5**。

如果减小阈值**T**，则正类的召回率升高，精度降低。

如果增加阈值**T**，则正类的召回率降低，精度增加。

1.准确率: $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{TN} + \text{FN})$

2. 正样本精确度: $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$ 表示的是召回为正样本的样本中，有多少是正样本

3.正样本召回率: $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$ 表示有多少样本被召回类，通常召回率高，精确度低

4.F1 score: $\text{F1 score} = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall})$

有时不仅关注正样本的准确率，也关心其召回率，但是又不想用**Accuracy**来进行衡量，一个折中的指标是采用**F-score**

评估指标（二分问题）

5. ROC曲线与AUC指标

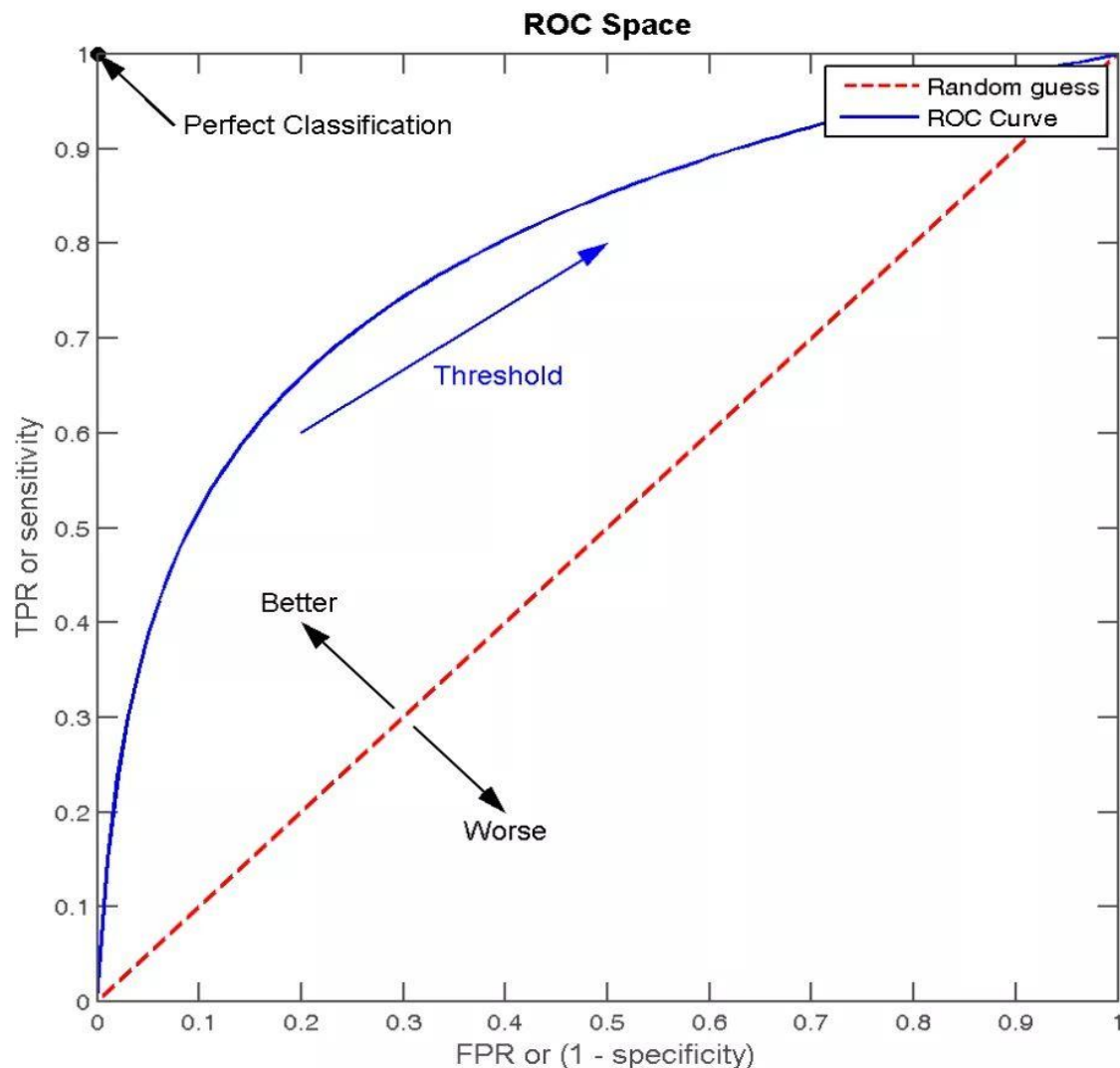
横坐标是false positive rate(FPR)，纵坐标是true positive rate(TPR)， $TPR = TP / (TP + FN)$ ，代表分类器预测的正类中实际正实例占有所有正实例的比例， $FPR = FP / (FP + TN)$ ，代表分类器预测的正类中实际负实例占有所有负实例的比例

ROC曲线越接近左上角，该分类器的性能越好。

测试集中的正负样本分布变化时，ROC曲线能保持不变，不均衡样本问题通常选ROC作为评价标准。

如果我们想通过两条ROC曲线来定量评估两个分类器的性能，就可以使用AUC指标。

AUC（Area Under Curve）为ROC曲线下的面积，随机挑选一个正样本以及一个负样本，AUC表征的就是有多大的概率，分类器会对正样本给出的预测值高于负样本



常见问题最后一层激活函数与损失函数的选择

问题类型	最后一层激活	损失函数
二分类问题	sigmoid	binary_crossentropy
多分类、单标签问题	softmax	categorical_crossentropy
多分类、多标签问题	sigmoid	binary_crossentropy
回归到任意值	无	mse
回归到0-1范围内的值	sigmoid	mse或binary_crossentropy

参考文献:

1. <https://blog.csdn.net/ericcchen/article/details/80102025>
2. https://blog.csdn.net/cymy001/article/details/78649132?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase
3. https://blog.csdn.net/ericcchen/article/details/80102006?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.nonecase
4. <http://yann.lecun.com/exdb/publis/pdf/chopra-05.pdf>
5. <http://tflearn.org/objectives/#weak-crossentropy-2d>
6. <http://tflearn.org/objectives/#roc-auc-score>
7. <https://zhuanlan.zhihu.com/p/59481933>