尚硅谷大模型技术之深度学习面试题

（作者：尚硅谷研究院）

版本：V0.9.0

1. 神经网络
   1. 什么是深度学习，它与传统机器学习有什么不同？

深度学习是机器学习的一个子集，涉及训练人工神经网络来执行通常由人类完成的任务，如图像和语音识别、自然语言处理和决策。它与传统机器学习的不同之处在于，它使用多层人工神经元，即深度神经网络，从大型复杂数据集中提取和处理信息。

* 1. 权重初始化如何影响深度学习模型的性能？

权重初始化是训练深度学习模型的重要步骤，因为它决定了优化过程的起点。如果用小的随机值初始化权重，模型将很难从数据中学习，因为梯度太小了。另一方面，如果权重初始化随机值较大，则模型收敛速度较快，但可能无法找到损失函数的全局最小值。选择合适的权值初始化方法，如 He 初始化或Glorot 初始化，有助于提高模型的性能。

权重初始化的目的是在深度神经网络中前向传递时，阻止网络层的激活函数输出爆炸（无穷大）或者消失（0）。如果网络层的输出爆炸或者消失，损失函数的梯度也会变得很大或者很小，无法有效后向传递，使得神经网络需要更长的时间才能收敛甚至无法收敛。

初始化方法有：Xavier初始化、Kaiming初始化、随机初始化

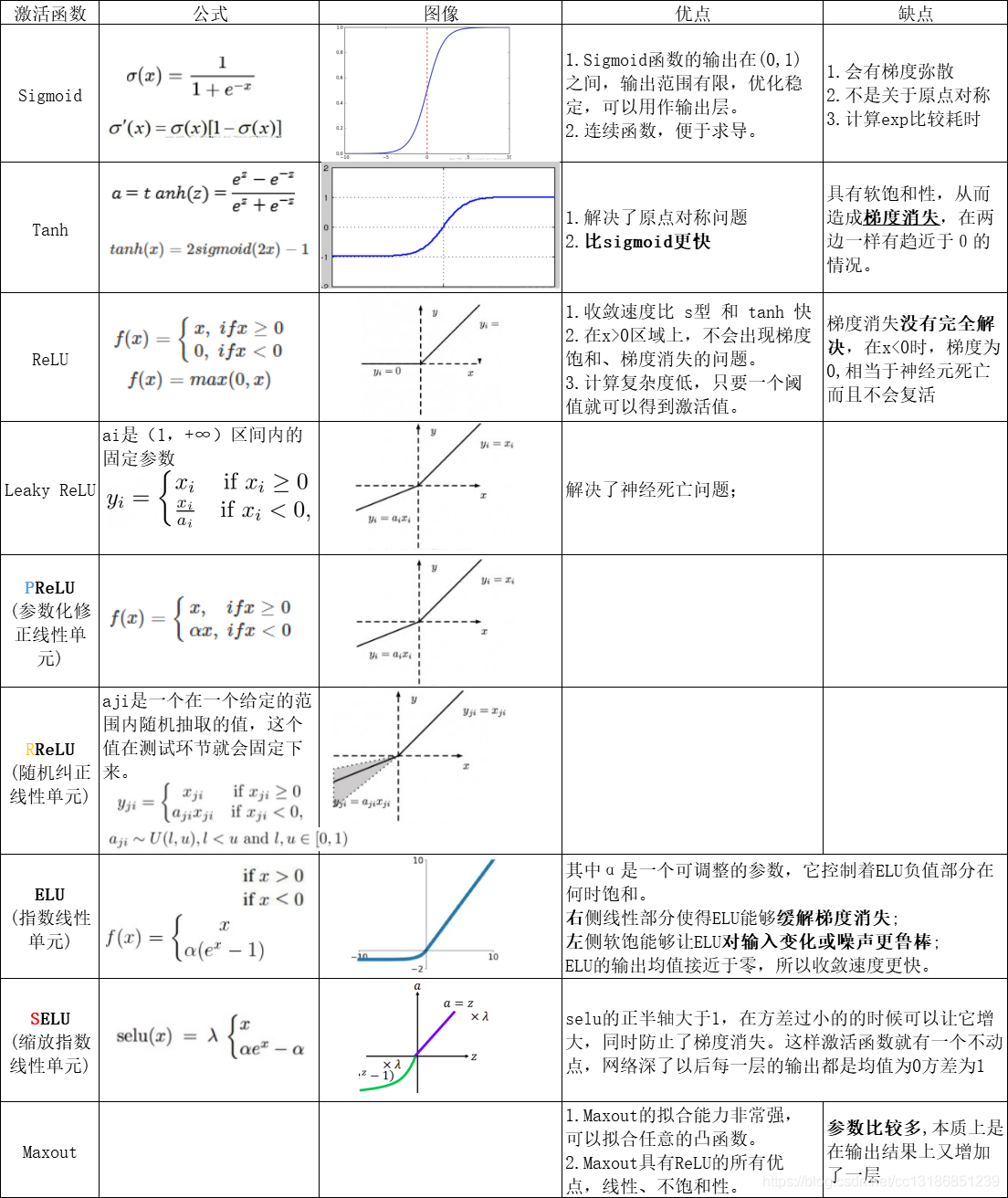
* 1. 你能解释感知器和 S 形神经元的区别吗？

感知器是使用线性激活函数的单层神经网络，用于二元分类。sigmoid 神经元是使用 sigmoid 激活函数的单层神经网络，可用于二元分类和多类分类。

* 1. 激活函数在神经网络中的作用是什么？

神经网络中的激活函数是用来将非线性引入网络的。如果没有非线性，神经网络将只是一个线性模型，无法学习数据中的复杂关系。激活函数，如sigmoid、ReLU 和 tanh，通过将输入压缩到一定范围内(如 0 到 1 或-1 到 1)，将非线性引入网络。

* 1. 各个激活函数的优缺点？



* 1. 为什么ReLU常用于神经网络的激活函数？

1.在**前向传播**和**反向传播**过程中，ReLU相比于Sigmoid等激活函数计算量小；

2**.避免梯度消失**问题。对于深层网络，Sigmoid函数反向传播时，很容易就会出现梯度消失问题（在Sigmoid接近饱和区时，变换太缓慢，导数趋于0，这种情况会造成信息丢失），从而无法完成深层网络的训练。

3.可以**缓解过拟合问题**的发生。Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生。

4.相比Sigmoid型函数，ReLU函数有助于随机梯度下降方法收敛。

**为什么需要激活功能？**

激活函数是用来加入非线性因素的，因为线性模型的表达能力不够。

* 1. 梯度消失和梯度爆炸的解决方案？梯度爆炸引发的问题？

梯度消失：靠近输出层的hidden layer 梯度大，参数更新快，所以很快就会收敛；

而靠近输入层的hidden layer 梯度小，参数更新慢，几乎就和初始状态一样，随机分布。

另一种解释：当**反向传播**进行很多层的时候，由于每一层都对前一层梯度乘以了一个小数，因此越往前传递，梯度就会越小，训练越慢。

梯度爆炸：前面layer的梯度通过训练变大，而后面layer的梯度指数级增大。

**①**在**深度多层感知机**(MLP)网络中，梯度爆炸会引起网络不稳定，最好的结果是无法从训练数据中学习，而最坏的结果是出现无法再更新的 NaN 权重值。

**②**在**RNN**中，梯度爆炸会导致网络不稳定，无法利用训练数据学习，最好的结果是网络无法学习长的输入序列数据。

手机屏幕截图

描述已自动生成

* 1. 如何确定是否出现梯度爆炸？

模型不稳定，导致更新过程中的损失出现显著变化；

训练过程中模型梯度快速变大；

训练过程中模型权重变成 NaN 值；

训练过程中，每个节点和层的误差梯度值持续超过 1.0。

* 1. 如何判断模型是否欠拟合

神级网络欠拟合的特征就是模型训练了足够长但时间后, loss 值依然很大甚至与初始值没有太大区别，且精度很低，测试集亦如此。根据我的总结，原因一般有以下几种：

神经网络的拟合能力不足；

网络配置的问题；

数据集配置的问题；

训练方法出错（初学者经常碰到，原因千奇百怪）。

* 1. 如何防止神经网络的过拟合？

当一个模型在训练数据上训练得太好，而在新的、看不见的数据上表现不佳时，就会出现过拟合。有几种技术可用于防止神经网络中的过拟合，如正则化、早期停止、dropout 和交叉验证。正则化包括在损失函数中添加惩罚项，以阻止网络给某些神经元分配大的权重。早期停止包括监视网络在验证集上的性能，并在性能开始趋于稳定时停止训练。Dropout 包括在训练过程中随机关闭一定比例的神经元，以减少网络对任何特定神经元的依赖。交叉验证包括将数据分成多个子集，并在不同的子集上训练和评估模型，以便更好地估计模型在未见数据上的性能。

* 1. 如何决定神经网络的层数和神经元数？

神经网络中的层数和神经元的数量被称为网络的架构。网络的架构通常是通过试验和错误的过程来确定的。一种常见的方法是从少量的层和神经元开始，逐渐增加它们，直到网络在验证集上的性能停止提高。另一种方法是使用网格搜索和随机搜索等技术系统地探索不同的体系结构，并找到性能最好的体系结构。

* 1. 神经网络中有哪些正则化技术？

L2正则化（**Ridge**）; L1正则化（**Lasso**）;

权重衰减; 丢弃法;

批量归一化; 数据增强

早停法

* 1. 批量归一化(BN) 如何实现？作用？

实现**过程**: 计算训练阶段mini\_batch数量激活函数前结果的**均值**和**方差**，然后对其进行**归一化**，最后对其进行缩放和平移。

**作用**： 1.可以使用更高的**学习率**进行优化；

2.**移除**或使用较低的 **dropout**；

3.**降低L2权重**衰减系数；

4.调整了数据的分布，不考虑激活函数，它让每一层的输出归一化到了均值为0方差为1的分布，这保证了梯度的有效性，可以解决反向传播过程中的梯度问题。

* 1. 神经网络中权值共享的理解？

权值共享这个词是由**LeNet5**模型提出来的。以**CNN**为例，在对一张图偏进行卷积的过程中，使用的是同一个卷积核的参数。

比如一个3×3×1的卷积核，这个卷积核内9个的参数被整张图共享，而不会因为图像内位置的不同而改变卷积核内的权系数。

通俗说：就是用一个卷积核不改变其内权系数的情况下卷积处理整张图片。

* 1. 对fine-tuning(微调模型)的理解？为什么要修改最后几层神经网络权值？

使用预训练模型的好处：在于利用训练好的SOTA模型权重去做特征提取，可以节省我们训练模型和调参的时间。

理由：

1.CNN中更靠近底部的层（定义模型时先添加到模型中的层）编码的是更加通用的可复用特征，而更靠近顶部的层（最后添加到模型中的层）编码的是更专业化的特征。微调这些更专业化的特征更加有用，它更代表了新数据集上的有用特征。

2.训练的参数越多，过拟合的风险越大。很多SOTA模型拥有超过千万的参数，在一个不大的数据集上训练这么多参数是有过拟合风险的，除非你的数据集像Imagenet那样大。

* 1. 什么是Dropout？为什么有用？它是如何工作的？

**背景**：如果模型的参数太多，数据量又太小，则容易产生过拟合。为了解决过拟合，就同时训练多个网络。然后多个网络取均值。费时！

**介绍**：Dropout可以防止过拟合，在前向传播的时候，让某个神经元的激活值以一定的概率 P停止工作，这样可以使模型的泛化性更强。

Dropout效果跟**bagging**效果类似（bagging是减少方差variance，而boosting是减少偏差bias）。

加入dropout会使神经网络训练时间长，模型预测时不需要dropout，记得关掉。‍

**具体流程**：

i.随机删除（临时）网络中一定的隐藏神经元，输入输出保持不变，

ii.让输入通过修改后的网络。然后把得到的损失同时修改后的网络进行反向传播。在未删除的神经元上面进行参数更新

iii.重复该过程（恢复之前删除掉的神经元，以一定概率删除其他神经元。前向传播、反向传播更新参数）

* 1. 如何选择dropout 的概率？

**input** 的 dropout 概率推荐是 0.8， **hidden layer** 推荐是0.5

为什么dropout可以解决过拟合？

1.取平均的作用:

Dropout产生了许多子结构之后的操作，父神经网络有N个节点，加入Dropout之后可以看做在权值不变的情况下（参数共享）将模型数量扩增到指数级别

2.减少神经元之间复杂的共适应关系，迫使网络去学习更加鲁棒；

Dropout 在训练和测试的区别？

训练时随机删除一些神经元，在测试模型时将所有的神经元加入。

* 1. 什么是Adam？Adam和SGD之间的主要区别是什么？
  2. 除了SGD和Adam之外，你还知道哪些优化算法？

主要有三大类：

a.基本梯度下降法，包括GD，BGD，SGD；

b.动量优化法，包括Momentum，NAG等；

c.自适应学习率优化法，包括Adam，AdaGrad，RMSProp等。

* 1. 一阶优化和二阶优化的方法有哪些？为什么不使用二阶优化？

一阶优化方法有：

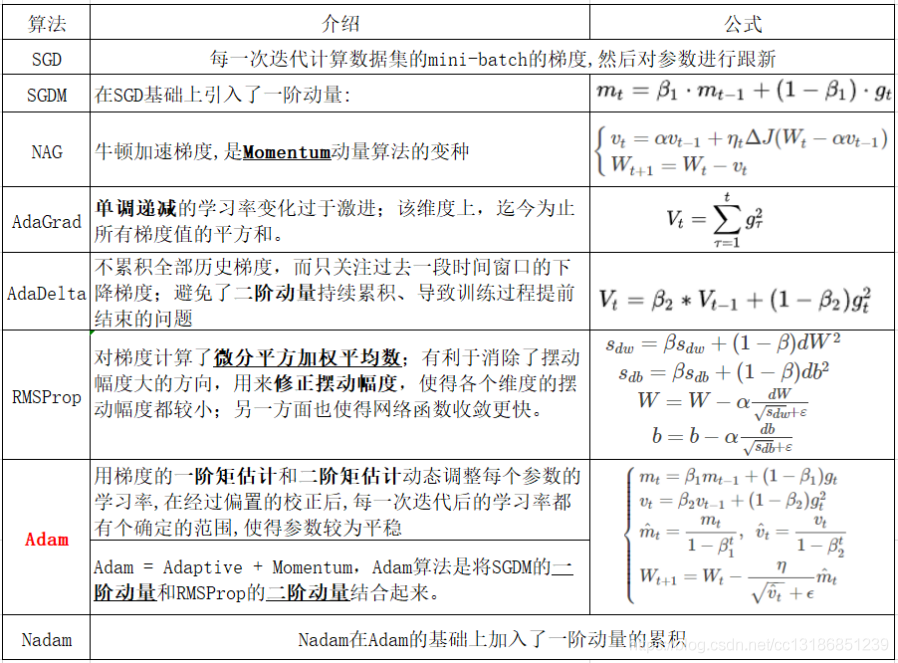
SGD -> SGDM ->NAG -> AdaGrad -> AdaDelta / RMSProp(加速梯度下降) -> Adam -> Nadam

二阶优化

定义：对目标函数进行二阶泰勒展开，也就是二阶梯度优化方法

牛顿法 + BFGS法

利用二阶泰勒展开,即牛顿法需要计算**Hessian**矩阵的逆，计算量大，在深度学习中并不常用。因此一般使用的是一阶梯度优化。



* 1. 为什么Momentum可以加速训练？

动量其实**累加了历史梯度更新方向**，所以在每次更新时，要是当前时刻的梯度与历史时刻梯度方向相似，这种趋势在当前时刻则会加强；要是不同，则当前时刻的梯度方向减弱。动量方法限制了梯度更新方向的随机性，使其沿正确方向进行。

* 1. batch size和epoch的平衡

手机屏幕截图

描述已自动生成

* 1. 什么时候使用Adam和SGD？

Adam等自适应学习率算法对于**稀疏数据**具有优势，且收敛速度很快；

但精调参数的SGD（+Momentum）往往能够取得更好的最终结果

Adam+SGD 组合策略：先用Adam快速下降，再用SGD调优。

如果模型是非常稀疏的，那么优先考虑自适应学习率的算法；

在模型设计实验过程中，要快速验证新模型的效果，用**Adam**进行快速实验优化；

在模型上线或者结果发布前，可以用精调的**SGD**进行模型的极致优化。

* 1. 神经网络为什么不用拟牛顿法而是用梯度下降？

手机屏幕截图

描述已自动生成

**牛顿法**原理：使用函数f(x)的泰勒级数的前面几项来寻找方程f(x)= 0的根。

* 1. 学习率太大(太小)时会发生什么？如何设置学习率？‍

1.基于经验的手动调整

通过尝试不同的固定学习率，如3、1、0.5、0.1、0.05、0.01、0.005，0.0001等，观察迭代次数和loss的变化关系，找到loss下降最快关系对应的学习率。

太小，收敛过程将变得十分缓慢;

太大，梯度可能会在最小值附近来回震荡(loss爆炸)，甚至可能无法收敛。

2. 基于策略的调整

①fixed 、exponential、polynomial

②自适应动态调整。adadelta、adagrad、ftrl、momentum、rmsprop、sgd

文本, 信件

描述已自动生成Logit：对它取对数。 

* 1. BN和Dropout在训练和测试时的差别?

BN:

训练时：是对每一个batch的训练数据进行归一化，也即是用每一批数据的均值和方差。

测试时：都是对单个样本进行测试。这个时候的均值和方差是全部训练数据的均值和方差，这两个数值是通过移动平均法求得。

当一个模型训练完成之后，它的所有参数都确定了，包括均值和方差，gamma和bata。

Dropout:

只有在训练的时候才采用，是为了减少神经元对部分上层神经元的依赖，类似将多个不同网络结构的模型集成起来，减少过拟合的风险。

* 1. 若网络初始化为0的话有什么问题？

w初始化全为0，很可能直接导致**模型失效**，**无法收敛**。

* 1. sigmoid和softmax的区别？softmax的公式？

图片包含 图示

描述已自动生成

表格

中度可信度描述已自动生成

* 1. 改进的softmax损失函数有哪些？

1.Large Margin Softmax Loss (**L-Softmax**)

加了一个margin，要保证大于某一个阈值。这样可以尽量实现让类间间距更大化，类内距离更小。

2.**Center Loss**

通过将特征和特征中心的距离和softmax loss一同作为损失函数，使得类内距离更小，有点L1，L2正则化。

* 1. 深度学习调参有哪些技巧？

1.准备数据

1.1 保证有大量、高质量并且带有干净标签的数据;

1.2 样本要随机化，防止大数据淹没小数据;

1.3 样本要做归一化.

2.预处理：0均值和1方差化

3.minibatch：建议值128，不要用过大的数值，否则很容易过拟合

4.梯度归一化：其实就是计算出来梯度之后，要除以minibatch的数量。

5.学习率

5.1 用一个一般的lr开始，然后逐渐的减小它;

5.2 建议值是0.01，适用于很多NN的问题，一般倾向于小一点;

5.3 对于调度学习率的建议：如果在验证集上性能不再增加就让学习率除以2或者5，然后继续，学习率会一直变得很小，到最后就可以停止训练了。

5.4 很多人用的一个设计学习率的原则就是监测一个比率（范数和权值范数之间的比率），如果这个比率在10-3附近，如果小于这个值，学习会很慢，如果大于这个值，那么学习很不稳定，由此会带来失败。

6.使用验证集，可以知道什么时候开始降低学习率，和什么时候停止训练。

7.权重初始化的选择：

7.1 用高斯分布乘上一个很小的数

7.2 在深度网络中，随机初始化权重，使用SGD的话一般处理的都不好，这是因为初始化的权重太小了。这种情况下对于浅层网络有效，但是当足够深的时候就不行了，因为weight更新的时候，是靠很多weight相乘的，越乘越小，有点类似梯度消失。

8.如果训练RNN或者LSTM，务必保证gradient的norm被约束在15或者5（前提还是要先归一化gradient），这一点在RNN和LSTM中很重要。

9.Adam收敛速度的确要快一些，可结果往往没有sgd + momentum的解好（如果模型比较复杂的话，sgd是比较难训练的，这时候用adam较好）

10.如果使用LSTM来解决长时依赖的问题，记得初始化bias的时候要大一点

11.尽可能想办法多的扩增训练数据，如果使用的是图像数据，不妨对图像做一点扭转之类的操作，来扩充数据训练集合。

12.使用dropout。

1.early stop，发现val\_loss没更新，就尽早停止。

2.评价最终结果的时候，多做几次，然后平均一下他们的结果。

* 1. SGD每步做什么，为什么能online learning？

online learning强调的是学习是实时的，流式的，每次训练不用使用全部样本，而是以之前训练好的模型为基础，每来一个样本就更新一次模型，这种方法叫做OGD（online gradient descent）,目的是快速地进行模型的更新，提升模型时效性。而SGD的思想正是在线学习的思想。

* 1. 神经网络调参，要往哪些方向想？

损失函数的选择是否是合适的

学习率的选取是否合适

batch size选择是否合适

训练样本是否正常，是否需要增强

是否有设置batch normlaztion(数据归一化)

激活函数的类型需要选取恰当

选取合适的优化算法（例如梯度下降，Adam等）

是否存在过拟合

* 1. 深度学习训练中是否有必要使用L1获得稀疏解?

没必要。对于加了L1正则的神经网络，大部分深度学习框架自带的优化器训练获得不了稀疏解；如果稀疏解的模式是无规律的，这种稀疏意义不大。

* 1. 神经网络数据预处理方法有哪些？

中心化/零均值，归一化

* 1. 缺失值如何处理

在神经网络中处理缺失的数据可能是一个挑战。一种常见的方法是简单地删除包含缺失数据的所有样本，但这可能导致信息丢失。另一种方法是使用诸如平均值估算、中位数估算或多重估算等技术来估算缺失值。更复杂的方法是使用神经网络根据其他可用信息预测缺失值。

* 1. 如何初始化神经网络的权重？神经网络怎样进行参数初始化？

a:将所有的参数初始化为0

b:对w随机初始化

c: Xavier初始化：尽可能的让输入和输出服从相同的分布，这样就能够避免后面层的激活函数的输出值趋向于0

d: 何氏初试法

如果权重初始化太小，会导致神经元的输入过小，随着层数的不断增加，会出现信号消失的问题；也会导致sigmoid激活函数丢失非线性的能力，因为在0附近sigmoid函数近似是线性的。如果参数初始化太大，会导致输入状态太大。

对sigmoid激活函数来说，激活函数的值会变得饱和，从而出现梯度消失的问题。

* 1. 为什么构建深度学习模型需要使用GPU？

VGG16（在DL应用中经常使用16个隐藏层的卷积神经网络）大约具有1.4亿个参数，又称权重和偏见。如果用传统的方法，训练这种系统需要几年的时间。

神经网络的计算密集部分由多个矩阵乘法组成，可以简单地通过同时执行所有操作，而不是一个接一个地执行，要使用GPU（图形处理单元）而不是CPU来训练神经网络。

* 1. 前馈神经网络(FNN),递归神经网络(RNN)和 CNN 区别?

FNN 是指在一个vector内部的不同维度之间进行一次信息交互。

CNN是以逐次扫描的方式在一个vector的局部进行多次信息交互。

RNN就是在hidden state上增加了loop，看似复杂，其实就是从时间维度上进行了展开。是在不同时刻的多个vectors之间进行信息交互。

* 1. 神经网络可以解决哪些问题？

手机屏幕截图

描述已自动生成

* 1. 如何提高小型网络的精度？

1.模型蒸馏技术；

2.利用AutoML进行网络结构的优化。

* 1. 列举你所知道的神经网络中使用的损失函数

欧氏距离，交叉熵，对比损失，合页损失

* 1. 什么是鞍点问题？

梯度为0，海森矩阵不定的点，不是极值点。

* 1. 你能解释一下深度学习中迁移学习的概念吗？

迁移学习是使用预训练的神经网络作为新任务的起点的过程。它涉及到已经在一个大数据集上训练过的网络的权重，并将它们用作在一个较小的数据集上训练的新网络的初始权重。当特定任务的数据不足，或者新任务与原始任务相似时，这可能很有用。通过使用预训练好的网络作为起点，新网络可以从原网络学习到的特征中学习，因此需要更少的数据和更少的训练时间来达到良好的性能。

* 1. 如何评估一个深度学习模型的性能？

评估深度学习模型的性能可能具有挑战性，因为它取决于特定的任务和数据的特征。用于评估深度学习模型性能的常用指标包括准确性、精密度、召回率、F1 分数和 ROC-AUC。另一种常用的方法是在未见过的测试数据上评估模型，并与其他模型进行性能比较。考虑模型的业务影响以及它满足最终用户需求的程度也很重要。

* 1. 深度信念网络和深度神经网络的区别是什么？

深度信念网络(deep belief network, DBN)是一种由多层受限玻尔兹曼机(restricted Boltzmann machines, rbm)组成的生成模型。dbn 通常以贪婪的分层方式进行训练，其中每一层都被训练以重建前一层的激活。深度神经网络(dnn)是具有多层的前馈神经网络，可以使用反向传播等技术进行训练。虽然 DBN 和 DNN 都是深度架构，但 DBN 是生成模型，DNN 是判别模型。

* 1. 你能解释一下深度学习中强化学习的概念吗

强化学习是一种学习类型，其中代理通过执行动作和接受奖励或惩罚来学习在环境中做出决策。智能体的目标是学习一种策略，使期望的累积奖励随着时间的推移最大化。这可以通过使用 Q 学习和策略梯度等技术来实现，这些技术涉及训练神经网络来近似值函数或策略。

* 1. 你能解释一下深度学习中注意机制的概念吗？

注意机制是一种用于关注神经网络输入的特定部分的技术。它允许模型选择性地关注输入的不同部分，例如句子中的不同单词或图像中的不同区域，并以不同的方式衡量每个部分的重要性。注意机制广泛应用于机器翻译、图像字幕和其他涉及顺序数据或多模态数据的任务中。

* 1. 神经网络的深度和宽度分别指的是什么？

神经网络的深度决定了网络的表达能力，早期的backbone设计都是直接堆叠卷积层，它的深度指的是神经网络的层数；后来的backbone设计采用了更高效的module（或block）堆叠的方式，每个module是由多个卷积层组成，这时深度指的是module的个数。

神经网络的宽度决定了网络在某一层学习到的信息量，指的是卷积神经网络中最大的通道数，由卷积核数量最多的层决定。通常的结构设计中卷积核的数量随着层数越来越多的，直到最后一层featuremap达到最大，这是因为越到深层，featuremap的分辨率越小，所包含的信息越高级，所以需要更多的卷积核来进行学习。通道越多效果越好，但带来的计算量也会大大增加，所以具体设定也是一个调参的过程，并且各层通道数会按照8×的倍数来确定，这样有利于GPU的并行计算。

* 1. 模型的参数量指的是什么？怎么计算？

参数量指的是网络中可学习变量的数量，包括卷积核的权重weights，批归一化（BN）的缩放系数γ，偏移系数β，有些没有BN的层可能有偏置bias，这些都是可学习的参数，即在模型训练开始前被赋予初值，在训练过程根据链式法则不断迭代更新，整个模型的参数量主要是由卷积核的权重weights的数量决定，参数量越大，则该结构对平台运行的内存要求越高。

Kh×Kw×Cin×Cout（Conv卷积网络）

Cin×Cout（FC全连接网络）

* 1. 模型的FLOPs（计算量）指的是什么？怎么计算？

神经网络的前向推理过程基本上都是乘累加计算，所以它的计算量也是指的前向推理过程中乘加运算的次数，通常用FLOPs来表示，即floatingpointoperations(浮点运算数)。计算量越大，在同一平台上模型运行延时越长，尤其是在移动端/嵌入式这种资源受限的平台上想要达到实时性的要求就必须要求模型的计算量尽可能地低，但这个不是严格成正比关系，也跟具体算子的计算密集程度(即计算时间与IO时间占比)和该算子底层优化的程度有关。

Kh×Kw×Cin×Cout×H×W=params×H×W（Conv卷积网络）

CinxCout（FC全连接网络）

计算量=输出的featuremap\*当前层filter即（H×W×Cout）×（K×K×Cin）

* 1. 神经网络中Addition/Concatenate区别是什么？

Addition和Concatenate分支操作统称为shortcut，Addition是在ResNet中提出，两个相同维度的featuremap相同位置点的值直接相加，得到新的相同维度featuremap，这个操作可以融合之前的特征，增加信息的表达，Concatenate操作是在Inception中首次使用，被DenseNet发扬光大，和addition不同的是，它只要求两个featuremap的HW相同，通道数可以不同，然后两个featuremap在通道上直接拼接，得到一个更大的featuremap，它保留了一些原始的特征，增加了特征的数量，使得有效的信息流继续向后传递。

* 1. 如果在网络初始化时给网络赋予0的权重，这个网络能正常训练嘛

不能，因为初始化权重是0，每次传入的不同数据得到的结果是相同的。网络无法更新

* 1. 为什么神经网络种常用relu作为激活函数

在前向传播和反向传播过程中，ReLU相比于Sigmoid等激活函数计算量小；

在反向传播过程中，Sigmoid函数存在饱和区，若激活值进入饱和区，则其梯度更新值非常小，导致出现梯度消失的现象。而ReLU没有饱和区，可避免此问题；

ReLU可令部分神经元输出为0，造成网络的稀疏性，减少前后层参数对当前层参数的影响，提升了模型的泛化性能

* 1. ReLU函数在0处不可导，为什么还能用？

反馈神经网络正常工作需要的条件就是每一个点提供一个方向，即导数；0值不可微，本质上来说是因为这个地方可画多条切线，但我们需要的只是一条；由于这出现的0值的概率极低，任意选择一个子梯度就OK了，在0处的次微分集合是【0，1】；即选择其中一个就OK了；一般默认是0；

* 1. 数据不平衡的解决方法

欠采样-随机删除观测数量足够多的类，使得两个类别间的相对比例是显著的。虽然这种方法使用起来非常简单，但很有可能被我们删除了的数据包含着预测类的重要信息。o过采样-对于不平衡的类别，我们使用拷贝现有样本的方法随机增加观测数量。理想情况下这种方法给了我们足够的样本数，但过采样可能导致过拟合训练数据。

合成采样（SMOTE）-该技术要求我们用合成方法得到不平衡类别的观测，该技术与现有的使用最近邻分类方法很类似。问题在于当一个类别的观测数量极度稀少时该怎么做。比如说，我们想用图片分类问题确定一个稀有物种，但我们可能只有一幅这个稀有物种的图片。

在loss方面，采用focalloss等loss进行控制不平衡样本。

* 1. 训练过程中,若一个模型不收敛,那么是否说明这个模型无效?导致模型不收敛的原因有哪些?

并不能说明这个模型无效,导致模型不收敛的原因可能有数据分类的标注不准确,样本的信息量太大导致模型不足以fit整个样本空间。学习率设置的太大容易产生震荡,太小会导致不收敛。可能复杂的分类任务用了简单的模型。数据没有进行归一化的操作。

* 1. 为什么交叉熵可以用作代价函数

从数学上来理解就是，为了让学到的模型分布更接近真实数据的分布，我们需要最小化模型数据分布与训练数据之间的 KL 散度 ，而因为训练数据的分布是固定的，因此最小化 KL 散度等价于最小化交叉熵，而且交叉熵计算更简单，所以机器/深度学习中常用交叉熵 cross-entroy 作为分类问题的损失函数。

* 1. 什么样的数据集不适合深度学习

数据集太小，数据样本不足时，深度学习相对其它机器学习算法，没有明显优势。

数据集没有局部相关特性，目前深度学习表现比较好的领域主要是图像／语音／自然语言处理等领域，这些领域的一个共性是局部相关性。图像中像素组成物体，语音信号中音位组合成单词，文本数据中单词组合成句子，这些特征元素的组合一旦被打乱，表示的含义同时也被改变。对于没有这样的局部相关性的数据集，不适于使用深度学习算法进行处理。举个例子：预测一个人的健康状况，相关的参数会有年龄、职业、收入、家庭状况等各种元素，将这些元素打乱，并不会影响相关的结果。

* 1. HOG算法原理描述

方向梯度直方图（Histogram of Oriented Gradient, HOG）特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。在深度学习取得成功之前，Hog特征结合SVM分类器被广泛应用于图像识别中，在行人检测中获得了较大的成功。

HOG的核心思想是所检测的局部物体外形能够被光强梯度或边缘方向的分布所描述。通过将整幅图像分割成小的连接区域（称为cells），每个 cell 生成一个方向梯度直方图或者 cell 中pixel 的边缘方向，这些直方图的组合可表示（所检测目标的目标）描述子。

为改善准确率，局部直方图可以通过计算图像中一个较大区域(称为block)的光强作为 measure 被对比标准化，然后用这个值(measure)归一化这个 block 中的所有 cells 。这个归一化过程完成了更好的照射/阴影不变性。与其他描述子相比，HOG 得到的描述子保持了几何和光学转化不变性（除非物体方向改变）。因此HOG描述子尤其适合人的检测。

HOG 特征提取方法就是将一个image：

1. 灰度化（将图像看做一个x,y,z（灰度）的三维图像）

2. 划分成小 cells （2\*2）

3. 计算每个 cell 中每个 pixel 的 gradient（即 orientation）

4. 统计每个 cell 的梯度直方图（不同梯度的个数），即可形成每个 cell 的 descriptor。

* 1. HOG特征检测步骤

颜色空间归一化——>梯度计算——>梯度方向直方图——>重叠块直方图归一化———>HOG特征

图示

描述已自动生成

* 1. 移动端深度学习框架知道哪些，用过哪些？

知名的有TensorFlow Lite、小米MACE、腾讯的 ncnn 等，目前都没有用过

1. CNN

**2.1卷积神经网络的结构**

对图像（不同的数据窗口数据）和滤波矩阵做内积（逐个元素相乘再求和）的操作就是所谓的『卷积』操作。

**卷积神经网络**由输入层、卷积层、激励层、池化层、全连接层组成。

**①**最左边:

数据输入层，对数据做一些处理：

**去均值**（把输入数据各个维度都中心化为0，避免数据过多偏差，影响训练效果）

**归一化**（把所有的数据都归一到同样的范围）、PCA/白化等。CNN只对训练集做“去均值”这一步。

**②**中间是:

CONV：卷积层，线性乘积 求和。

RELU：激励层，ReLU是激活函数的一种。

POOL：**池化层**，即**取区域平均或最大**。



**③**最右边是:

FC：**全连接层**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

* 1. 卷积层和全连接层的区别是什么？

卷积层是局部连接，所以提取的是局部信息；全连接层是全局连接，所以提取的是全局信息；

当卷积层的局部连接是全局连接时，全连接层是卷积层的特例

* 1. 网络设计中，为什么卷积核设计尺寸都是奇数？

保证像素点中心位置，避免位置信息偏移

填充边缘时能保证两边都能填充，原矩阵依然对称

* 1. 池化层的作用

减小图像尺寸即数据降维，缓解过拟合，保持一定程度的旋转和平移不变性。

* 1. 阐述一下感受野的概念，并说一下在CNN中如何计算

感受野指的是卷积神经网络每一层输出的特征图上每个像素点映射回输入图像上的区域的大小，神经元感受野的范围越大表示其接触到的原始图像范围就越大，也就意味着它能学习更为全局，语义层次更高的特征信息，相反，范围越小则表示其所包含的特征越趋向局部和细节。因此感受野的范围可以用来大致判断每一层的抽象层次，并且我们可以很明显地知道网络越深，神经元的感受野越大。

卷积层的感受野大小与其之前层的卷积核尺寸和步长有关，与padding无关。

感受野在CNN中是呈指数级增加的。小卷积核（如 3\*3 ）通过多层叠加可取得与大卷积核（如 7\*7 ）同等规模的感受野，此外采用小卷积核有两个优势：

1. 小卷积核需多层叠加，加深了网络深度进而增强了网络容量( model capacity )和复杂度（ model complexity ）。

2. 增强了网络容量的同时减少了参数个数。

* 1. 卷积和池化操作的作用

1.局部感知，参数共享 的特点大大降低了网络参数，保证了网络的稀疏性。

2.通过卷积核的组合以及随着网络后续操作的进行，卷积操作可获取图像不同区域的不同类型特征；模型靠近底部的层提取的是局部的、高度通用的特征图，而更靠近顶部的层提取的是更加抽象的语义特征。

池化/汇合（ pooling ）操作作用如下：

1. 特征不变性（feature invariant）。汇合操作使模型更关注是否存在某些特征而不是特征具体的位置可看作是一种很强的先验，使特征学习包含某种程度自由度，能容忍一些特征微小的位移。

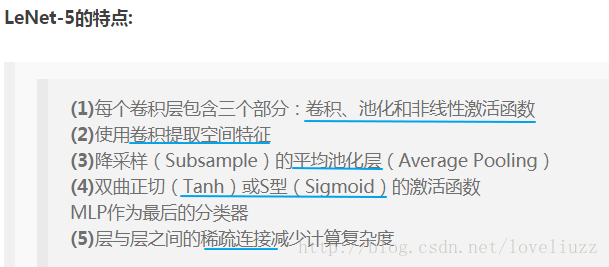
2. 特征降维。由于汇合操作的降采样作用，汇合结果中的一个元素对应于原输入数据的一个子区域（sub-region），因此汇合相当于在空间范围内做了维度约减（spatially dimension reduction），从而使模型可以抽取更广范围的特征。同时减小了下一层输入大小，进而减小计算量和参数个数。

3. 在一定程度上能防止过拟合（overitting），更方便优化。

* 1. 经典网络分类
     1. LeNet

最早用于数字识别;针对灰度图进行训练的，输入图像大小为**32\*32\*1**，5\*5卷积核，不包含输入层的情况下共有**7层**，每层都包含可训练参数。

输入的二维图像，先经过**两次卷积层**到**池化层**，再经过**全连接层**，最后使用**softmax**分类作为输出层。（conv1->pool->conv2->pool2再接全连接层）



形状

描述已自动生成

* + 1. AlexNet

用多层小卷积叠加来替换单个的大卷积。

输入尺寸：227\*227\*3

卷积层：5个

降采样层(池化层)：3个

全连接层：2个（不包含输出层）

输出层：1个。1000个类别

图示, 形状

描述已自动生成

AlexNet比LeNet表现更为出色的另一个原因是它使用了**ReLu**激活函数。

* + 1. AlexNet 对比LeNet 的优势？

1.AlexNet比LeNet更深；

2.用多层的小卷积来替换单个的大卷积；

3.非线性激活函数：ReLU

4.防止过拟合的方法：Dropout，数据增强

5.大数据训练：百万级ImageNet图像数据

6.其他：GPU实现，LRN归一化层的使用

* + 1. VGG

构筑了16~19层深的卷积神经网络，**VGG-16**中的16：含有参数的有16个层

VGGNet论文中全部使用了3\*3的小型卷积核和2\*2的最大池化层，通过不断加深网络结构来提升性能。

卷积层：CONV=3\*3 filters, s = 1, padding = same convolution。

池化层：MAX\_POOL = 2\*2 , s = 2。

优点：简化了卷积神经网络的结构；缺点：训练的特征数量非常大。

随着网络加深，图像的宽度和高度都在以一定的规律不断减小，每次池化后刚好缩小一半，信道数目不断增加一倍。

* + 1. VGG使用2个3\*3卷积的优势在哪里？

①减少网络层参数：

用两个3\*3卷积比用1个5\*5卷积拥有更少的参数量，只有后者的2\*3\*3/(5\*5)=0.72。但是起到的效果是一样的，两个33的卷积层串联相当于一个55的卷积层，感受野的大小都是5×5，即1个像素会跟周围5\*5的像素产生关联.

②更多的非线性变换：

2个33卷积层拥有比1个55卷积层更多的非线性变换（前者可以使用两次ReLU激活函数，而后者只有一次），使得卷积神经网络对特征的学习能力更强。

* + 1. 每层卷积是否只能用一种尺寸的卷积核？

可以，经典的神经网络一般都属于层叠式网络，每层仅用一个尺寸的卷积核，如**VGG**结构中使用了大量的3×3卷积层。

同一层特征图也可以分别使用多个不同尺寸的卷积核，以获得不同尺度的特征，再把这些特征结合起来，得到的特征往往比使用单一卷积核的要好。比如**GoogLeNet**、**Inception**系列的网络。

* + 1. Inception(GoogLeNet)

增加了卷积神经网络的宽度，在多个不同尺寸的卷积核上进行卷积后再聚合，并使用**1\*1**卷积降维减少参数量。

* + 1. Inception结构能不能缓解梯度消失？

可以，因为Inception结构额外计算了两个中间loss，防止了较深网络传播过程中的梯度消失问题。

* + 1. ResNet

**残差网络**解决了网络退化的问题（随着网络的深度增加，准确度反而下降了）

* + 1. ResNet为什么不用Dropout?

BN在训练过程对每个单个样本的forward均引入多个样本（Batch个）的统计信息，相当于自带一定噪音，起到正则效果，所以也就基本消除了Dropout的必要。（ResNet训练**152**层深的神经网络）

* + 1. ResNet网络越来越深，准确率会不会提升？

训练精度和测试精度迅速下降。

神经网络在反向传播过程中要不断地传播梯度，而当网络层数加深时，梯度在传播过程中会逐渐消失，导致无法对前面网络层的权重进行有效的调整。

* + 1. ResNet v1 与 ResNet v2的区别?

通过ResNet 残差学习单元的传播公式，发现前馈和反馈信号可以直接传输，

因此 捷径连接 的非线性激活函数（如ReLU）替换为 Identity Mappings。

同时，ResNet V2 在每一层中都使用了 Batch Normalization。这样处理之后，新的残差学习单元将比以前更容易训练且泛化性更强。

* + 1. DenseNet

含义：前面所有层与后面层的密集连接, **每一层的输入**都是前面所有层输出的并集，而该层所学习的特征图也会被直接传给其后面所有层作为输入

**优点**：缓解梯度消失问题，特征复用，加强特征传播，减少参数量

缺点：内存占用高

**梯度消失原因**：每一层都直接连接input和loss。

参数量少原因：每一层已经能够包含前面所有层的输出，只需要很少的特征图就可以了。

* + 1. DenseNet 比 ResNet 好？

1.ResNet连接方式可能会阻碍信息的流动，但是DenseNet每层的输出都和最终的输出直接相连，梯度可以直接从末端流到之前的所有的层。

2.DensetNet连接有**正则化**的作用，可以减少过拟合。

3.DenseNet直接**连接不同层的特征图**，而不是像ResNet一样element-wise sum。**2.2.6.2为什么 DenseNet 比 ResNet 更耗显存？**

DenseNet的特征图像比ResNet大很多，导致卷积过程的计算量比resnet大很多。

* 1. 卷积层有哪些基本参数？

①卷积核大小 (Kernel Size)：

定义了卷积的感受野 在过去常设为5，如LeNet-5；现在多设为3，通过堆叠3×33×3的卷积核来达到更大的感受域。

②卷积核步长 (Stride)：

常见设置为1，可以覆盖所有相邻位置特征的组合；当设置为更大值时相当于对特征组合降采样。

③填充方式 (Padding)

④输入通道数 ：指定卷积操作时卷积核的深度

⑤输出通道数 ：指定卷积核的个数

**感受野**：CNN每一层输出的特征图上的像素点在原始图像上映射的区域大小。

* 1. 如何计算卷积层的输出的大小？

IMG_256

K 是过滤器尺寸，P 是填充，S 是步幅

* 1. 如何计算卷积层参数数量？

卷积层参数量 = (filter size \* 前一层特征图的通道数)\* 当前层filter数量 + 当前层filter数量。（卷积核长度\*卷积核宽度\*通道数+1）\*卷积核个数

假设输入层矩阵维度是 96×96×3，第一层卷积层使用尺寸为 5×5、深度为 16 的过滤器（卷积核尺寸为 5×5、卷积核数量为 16），那么这层卷积层的参数个数为 5×5×3×16 + 16=1216个。

* 1. 有哪些池化方法？

池化操作也叫做**子采样**(Subsampling)或**降采样**(Downsampling)，往往会用在卷积层之后，通过池化来降低卷积层输出的特征维度，有效减少网络参数的同时还可以防止过拟合现象。

**①**最大池化 和 **②**平均池化

以最大池化为例，池化范围(2×2)(2×2)和滑窗步长(stride=2)(stride=2) 相同，仅提取一次相同区域的范化特征。

* 1. 池化层如何进行反向传播

池化层没有可以训练的参数，因此在卷积神经网络的训练中，池化层只需要将误差传递到上一层，并不需要做梯度的计算。要追求一个原则，那就是梯度之和不变。

* 1. 为什么maxpooling要更常用？什么场景下averagepooling比maxpooling更合适？

作用：对输入的特征图进行压缩，一方面使特征图变小，简化网络计算复杂度；一方面进行特征压缩，提取主要特征。

通常来讲，max-pooling的效果更好，虽然max-pooling和average-pooling都对数据做了下采样，但是max-pooling感觉更像是做了特征选择，选出了分类辨识度更好的特征，提供了非线性。pooling的主要作用一方面是去掉冗余信息，一方面要保留featuremap的特征信息，在分类问题中，我们需要知道的是这张图像有什么object，而不大关心这个object位置在哪，在这种情况下显然maxpooling比averagepooling更合适。在网络比较深的地方，特征已经稀疏了，从一块区域里选出最大的，比起这片区域的平均值来，更能把稀疏的特征传递下去。

average-pooling更强调对整体特征信息进行一层下采样，在减少参数维度的贡献上更大一点，更多的体现在信息的完整传递这个维度上，在一个很大很有代表性的模型中，比如说DenseNet中的模块之间的连接大多采用average-pooling，在减少维度的同时，更有利信息传递到下一个模块进行特征提取。

average-pooling在全局平均池化操作中应用也比较广，在ResNet和Inception结构中最后一层都使用了平均池化。有的时候在模型接近分类器的末端使用全局平均池化还可以代替Flatten操作，使输入数据变成一维向量。

* 1. 1\*1卷积的作用？

①加入非线性函数。卷积层之后经过激励层，提升网络的表达能力;

②对卷积核通道数进行降维和升维，减小参数量。

* 1. 卷积层和池化层有什么区别？



①卷积层有参数，池化层没有参数；

②经过卷积层节点矩阵深度会改变。池化层不会改变节点矩阵的深度，但是它可以缩小节点矩阵的大小。

* 1. 卷积核是否一定越大越好？

不一定，**缺点**：会导致计算量大幅增加，不利于训练更深层的模型，相应的计算性能也会降低。

卷积神经网络（**VGG**、**GoogLeNet**等），发现通过堆叠2个3×3卷积核可以获得与5×5卷积核相同的感受视野，同时参数量会更少（3×3×2+1 < $ 5×5×1+1$）

**优点**：

文本特征有时需要有较广的感受域让模型能够组合更多的特征（如词组和字符）

卷积核的大小并没有绝对的优劣，需要视具体的应用场景而定，但是极大和极小的卷积核都是不合适的，单独的1×1极小卷积核只能用作分离卷积而不能对输入的原始特征进行有效的组合，极大的卷积核通常会组合过多的无意义特征从而浪费了大量的计算资源。

* 1. 卷积在图像中有什么直观作用?

用来提取**图像的特征**，但不同层次的卷积操作提取到的特征类型是不相同的：

浅层卷积： 边缘特征

中层卷积： 局部特征

深层卷积： 全局特征

* 1. CNN中空洞卷积的作用是什么？

**空洞卷积**也叫扩张卷积，在保持参数个数不变的情况下增大了卷积核的感受野，同时它可以保证输出的特征映射的大小保持不变。一个扩张率为2的3×3卷积核，感受野与5×5的卷积核相同，但参数数量仅为9个。

* 1. 怎样才能减少卷积层参数量？

①使用堆叠小卷积核代替大卷积核：

VGG网络中2个3×3的卷积核可以代替1个5×5的卷积核

②使用分离卷积操作：

将原本K×K×C的卷积操作分离为K×K×1和1×1×C的两部分操作

③添加1×1的卷积操作：与分离卷积类似，但是通道数可变，在K×K×C1卷积前添加1×1×C2的卷积核（满足C2<C1）

④在卷积层前使用池化操作：池化可以降低卷积层的输入特征维度

* 1. 在进行卷积操作时，必须同时考虑通道和区域吗？

①标准卷积同时考虑通道和区域

②通道分离（深度分离）卷积网络(Xception网络):

首先对每一个通道进行各自的卷积操作，有多少个通道就有多少个过滤器。得到新的通道特征矩阵之后，再对这批新通道特征进行标准的1×1跨通道卷积操作。

* 1. 采用宽卷积,窄卷积的好处有什么？

宽卷积、窄卷积其实是一种**填充方式**。

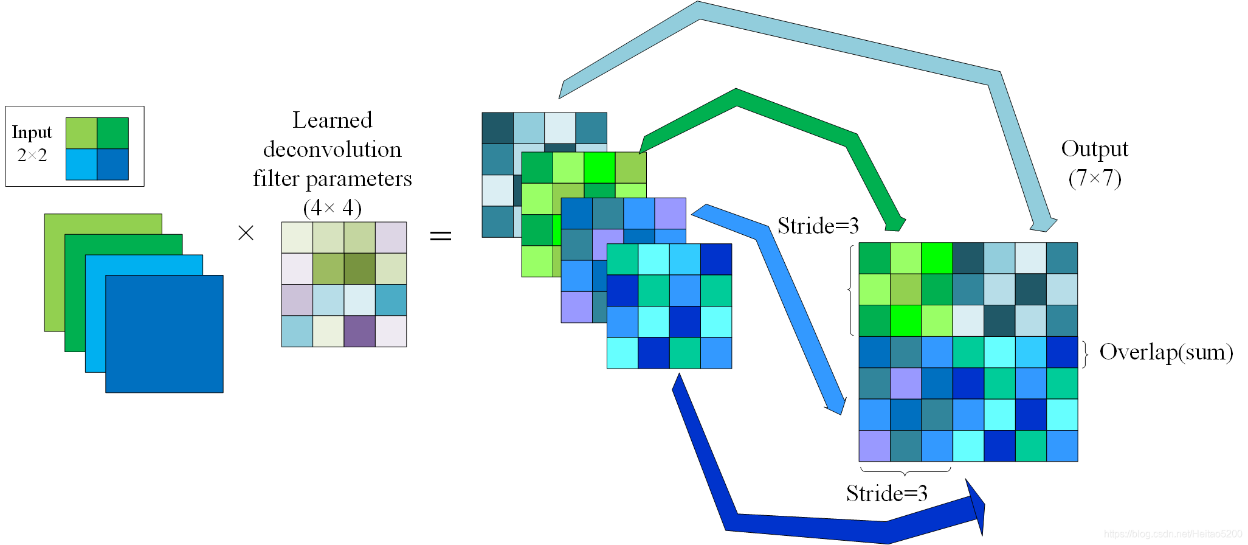
**①**宽卷积(**'SAME’**填充)：

对卷积核不满足整除条件的输入特征进行补全，以使卷积层的输出维度保持与输入特征维度一致。

**②**窄卷积(**'VALID’**填充):

不进行任何填充，在输入特征边缘位置若不足以进行卷积操作，则对边缘信息进行舍弃,因此在步长为1的情况下该填充方式的卷积层输出特征维度可能会略小于输入特征的维度。

* 1. 介绍反卷积(转置卷积)



正向传播时乘以卷积核的转置矩阵，反向传播时乘以卷积核矩阵，由卷积输出结果近似重构输入数据，**上采样**。

输入：2x2， 卷积核：4x4， 滑动步长：3， 输出：7x7

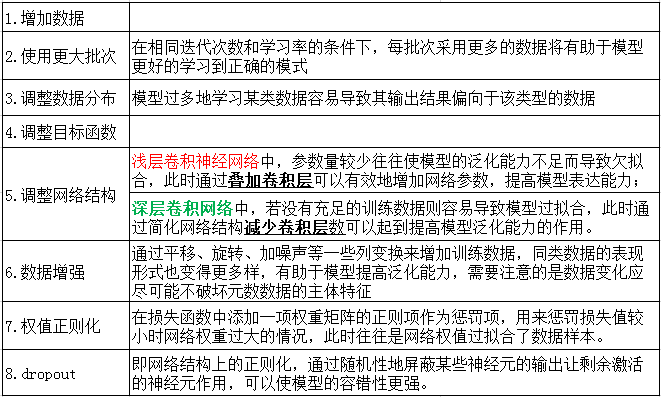
过程如下：

**①**输入图片每个像素进行一次**full卷积**，根据full卷积大小计算可以知道每个像素的卷积后大小为 1+4−1==4， 即4x4大小的特征图，输入有4个像素所以4个4x4的特征图。

**②**将4个特征图进行步长为3的相加； 输出的位置和输入的位置相同。步长为3是指每隔3个像素进行相加，重叠部分进行相加，即输出的第1行第4列是由红色特阵图的第一行第四列与绿色特征图的第一行第一列相加得到，其他类推。

可以看出反卷积的大小是由卷积核大小与滑动步长决定， in是输入大小， k是卷积核大小， s是滑动步长， out是输出大小 得到 **out=(in−1)\*s + k** 上图过程就是 (2 - 1) \* 3 + 4 = 7。

* 1. 如何提高卷积神经网络的泛化能力？



* 1. 卷积神经网络在NLP与CV领域应用的区别?

**自然语言处理**对一维信号（词序列）做操作，输入数据通常是离散取值（例如表示一个单词或字母通常表示为词典中的one hot向量）

**计算机视觉则**是对二维（图像）或三维（视频流）信号做操作。输入数据是连续取值（比如归一化到0，1之间的- 灰度值）。

* 1. 全连接、局部连接、全卷积与局部卷积的区别？



* 1. 卷积层和全连接层的区别？

1.卷积层是局部连接，所以提取的是局部信息；全连接层是全局连接，所以提取的是全局信息；

2.当卷积层的局部连接是全局连接时，全连接层是卷积层的特例；

* 1. Max pooling如何工作？还有其他池化技术吗？

1.Max pooling:选取滑动窗口的最大值

2.Average pooling：平均滑动串口的所有值

3.Global average pooling：平均每页特征图的所有值

* 1. 卷积神经网络的优点？为什么用小卷积核？

多个小的卷积核叠加使用要远比一个大的卷积核单独使用效果要好的多。

1.局部连接

这个是最容易想到的，每个神经元不再和上一层的所有神经元相连，而只和一小部分神经元相连。这样就减少了很多参数。

2.权值共享

一组连接可以共享同一个权重，而不是每个连接有一个不同的权重，这样又减少了很多参数。

3.下采样

Pooling层利用图像局部相关性的原理，对图像进行子抽样，可以减少数据处理量同时保留有用信息。通过去掉Feature Map中不重要的样本，进一步减少参数数量。

* 1. CNN拆成3x1 1x3的优点？

为了压缩模型参数量（这里参数由3x3=9降低到1x3+3x1=6），但是计算量基本没变（乘数目没变）。

* 1. BN、LN、IN、GN和SN的区别？

将输入的 feature map shape 记为[N, C, H, W]，其中N表示batch size，即N个样本；C表示通道数；H、W分别表示特征图的高度、宽度。

表格

描述已自动生成

* 1. 为什么需要卷积？不能使用全连接层吗？‍
  2. 为什么降采用使用max pooling，而分类使用average pooling?
  3. CNN是否抗旋转？如果旋转图像，CNN的预测会怎样？
  4. 什么是数据增强？为什么需要它们？你知道哪种增强？

数据增强是一种通过对现有数据应用一系列操作来人为地增加数据集大小的技术。这些操作可以包括随机裁剪、翻转、旋转、缩放和其他形式的图像处理。数据增强可以通过创建原始数据的新版本来帮助减少过拟合并改进模型的泛化。

翻转： Fliplr,Flipud 。不同于旋转180度，这是类似镜面的翻折，跟人在镜子中的映射类似，常用水平、上下镜面翻转。

旋转： rotate 。顺时针/逆时针旋转，最好旋转 90-180 度，否则会出现边缘缺失或者超出问题，如旋转 45 度。

缩放： zoom 。图像可以被放大或缩小，imgaug库可用Scal函数实现。

裁剪： crop 。一般叫随机裁剪，操作步骤是：随机从图像中选择一部分，然后降这部分图像裁剪出来，然后调整为原图像的大小。根本上理解，图像crop就是指从图像中移除不需要的信息，只保留需要的部分

平移： translation 。平移是将图像沿着x或者y方向（或者两个方向）移动。我们在平移的时候需对背景进行假设，比如说假设为黑色等等，因为平移的时候有一部分图像是空的，由于图片中的物体可能出现在任意的位置，所以说平移增强方法十分有用。

放射变换： Affine 。包含：平移( Translation )、旋转( Rotation )、放缩( zoom )、错切( shear )。添加噪声：过拟合通常发生在神经网络学习高频特征的时候，为消除高频特征的过拟合，可以随机加入噪声数据来消除这些高频特征。 imgaug 图像增强库使用 GaussianBlur 函数。

亮度、对比度增强：这是图像色彩进行增强的操作

锐化： Sharpen 。imgaug库使用Sharpen函数。

* 1. 离线数据增强和在线数据增强有什么区别?

数据增强分两类，一类是离线增强，一类是在线增强：

1. 离线增强 ：直接对硬盘上的数据集进行处理，并保存增强后的数据集文件。数据的数目会变成增强因子 x 原数据集的数目 ，这种方法常常用于数据集很小的时候

2. 在线增强 ：这种增强的方法用于，获得 batch 数据之后，然后对这个batch的数据进行增强，如旋转、平移、翻折等相应的变化，由于有些数据集不能接受线性级别的增长，这种方法长用于大的数据集，很多机器学习框架已经支持了这种数据增强方式，并且可以使用GPU优化计算。

* 1. 如何选择要使用的增强？‍
  2. 什么是迁移学习？它是如何工作的？‍
  3. 什么是目标检测？你知道有哪些框架吗？
  4. 什么是对象分割？你知道有哪些框架吗？
  5. 上采样的原理和常用方式

在卷积神经网络中，由于输入图像通过卷积神经网络(CNN)提取特征后，输出的尺寸往往会变小，而有时我们需要将图像恢复到原来的尺寸以便进行进一步的计算(如图像的语义分割)，这个使图像由小分辨率映射到大分辨率的操作，叫做上采样，它的实现一般有三种方式：

a.插值，一般使用的是双线性插值，因为效果最好，虽然计算上比其他插值方式复杂，但是相对于卷积计算可以说不值一提，其他插值方式还有最近邻插值、三线性插值等；

b.转置卷积又或是说反卷积，通过对输入featuremap间隔填充0，再进行标准的卷积计算，可以使得输出featuremap的尺寸比输入更大；

c.MaxUnpooling，在对称的maxpooling位置记录最大值的索引位置，然后在unpooling阶段时将对应的值放置到原先最大值位置，其余位置补0

* 1. 下采样的作用是什么？通常有哪些方式？

下采样层有两个作用，一是减少计算量，防止过拟合；二是增大感受野，使得后面的卷积核能够学到更加全局的信息。

下采样的方式主要有两种：

a.采用stride为2的池化层，如Max-pooling和Average-pooling，目前通常使用Max-pooling，因为他计算简单而且能够更好的保留纹理特征；

b.采用stride为2的卷积层，下采样的过程是一个信息损失的过程，而池化层是不可学习的，用stride为2的可学习卷积层来代替pooling可以得到更好的效果，当然同时也增加了一定的计算量。

* 1. 深度可分离卷积的概念和作用

深度可分离卷积将传统的卷积分两步进行，分别是depthwise和pointwise。首先按照通道进行计算按位相乘的计算，深度可分离卷积中的卷积核都是单通道的，输出不能改变featuremap的通道数，此时通道数不变；然后依然得到将第一步的结果，使用1\*1的卷积核进行传统的卷积运算，此时通道数可以进行改变。

计算量的前后对比：

Kh×Kw×Cin×Cout×H×W

变成了Kh×Kw×Cin×H×W+1×1×Cin×Cout×H×W

* 1. 神经网络中1×1卷积有什么作用？

降维，减少计算量；在ResNet模块中，先通过1×1卷积对通道数进行降通道，再送入3×3的卷积中，能够有效的减少神经网络的参数量和计算量；

升维；用最少的参数拓宽网络通道，通常在轻量级的网络中会用到，经过深度可分离卷积后，使用1×1卷积核增加通道的数量，例如mobilenet、shufflenet等；

实现跨通道的交互和信息整合；增强通道层面上特征融合的信息，在featuremap尺度不变的情况下，实现通道升维、降维操作其实就是通道间信息的线性组合变化，也就是通道的信息交互整合的过程；

1×1卷积核可以在保持featuremap尺度（不损失分辨率）不变的情况下，大幅增加非线性特性（利用后接的非线性激活函数）。

* 1. 增大感受野的方法

空洞卷积、池化操作、较大卷积核尺寸的卷积操作

* 1. 深度学习为什么在计算机视觉领域这么好？

传统的计算机视觉方法需首先基于经验手动设计特征，然后使用分类器分类，这两个过程都是分开的。而深度学习里的卷积网络可实现对局部区域信息的提取，获得更高级的特征，当神经网络层数越多时，提取的特征会更抽象，将更有助于分类，同时神经网路将提取特征和分类融合在一个结构中。

* 1. 什么是GroupConvolution

若卷积神将网络的上一层有N个卷积核,则对应的通道数也为N。设群数目为M,在进行卷积操作的时候,将通道分成M份,每个group对应N/M个通道,然后每个group卷积完成后输出叠在一起,作为当前层的输出通道。

* 1. CNN为什么比DNN在图像识别上更好

DNN的输入是向量形式，并未考虑到平面的结构信息，在图像和NLP领域这一结构信息尤为重要，例如识别图像中的数字，同一数字与所在位置无关（换句话说任一位置的权重都应相同），CNN的输入可以是tensor，例如二维矩阵，通过filter获得局部特征，较好的保留了平面结构信息。

* 1. CNN 权值共享问题

首先权值共享就是滤波器共享，滤波器的参数是固定的，即是用相同的滤波器去扫一遍图像，提取一次特征特征，得到feature map。在卷积网络中，学好了一个滤波器，就相当于掌握了一种特征，这个滤波器在图像中滑动，进行特征提取，然后所有进行这样操作的区域都会被采集到这种特征，就好比上面的水平线。

1. RNN
   1. 什么是RNN

核心思想：像人一样拥有记忆能力。用以往的记忆和当前的输入，生成输出。

RNN 和 传统神经网络 最大的区别:

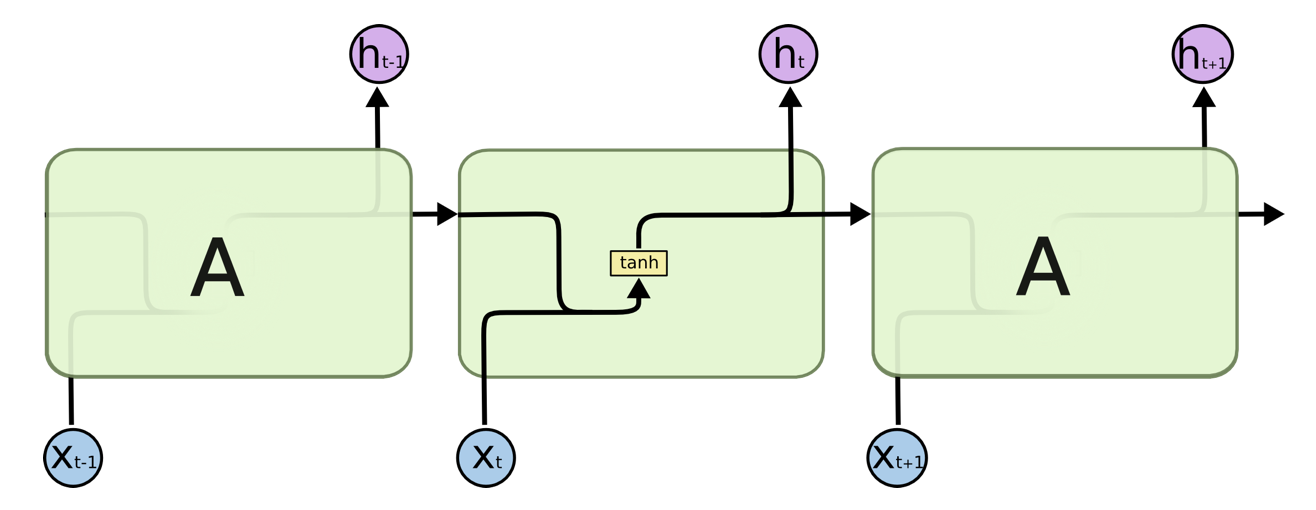
在于每次都会将前一次的输出结果，带到下一次的隐藏层中，一起训练。

图示, 文本

描述已自动生成

* 1. RNN应用场景

1.文本生成 2.语音识别 3.机器翻译 4.生成图像描述 5.视频标记



* 1. RNN缺点

RNN 有短期记忆问题，无法处理很长的输入序列

训练 RNN 需要投入极大的成本

RNN 是一种死板的逻辑，越晚的输入影响越大，越早的输入影响越小，且无法改变这个逻辑。

* 1. RNNs训练和传统ANN训练异同点？

相同点：都使用BP误差反向传播算法。

不同点：

RNNs网络参数W,U,V是共享的，而传统神经网络各层参数间没有直接联系。

对于RNNs，在使用梯度下降算法中，每一步的输出不仅依赖当前步的网络，还依赖于之前若干步的网络状态。

* 1. 为什么RNN 训练的时候Loss波动很大?

由于RNN特有的memory会影响后期其他的RNN的特点，梯度时大时小，lr没法个性化的调整，导致RNN在train的过程中，Loss会震荡起伏，为了解决RNN的这个问题，在训练的时候，可以设置临界值，当梯度大于某个临界值，直接截断，用这个临界值作为梯度的大小，防止大幅震荡。

* 1. RNN中为什么会出现梯度消失？

梯度消失现象：累乘会导致激活函数导数的累乘，如果取tanh或sigmoid函数作为激活函数的话，那么必然是一堆小数在做乘法，结果就是越乘越小。随着时间序列的不断深入，小数的累乘就会导致梯度越来越小直到接近于0，这就是“梯度消失“现象。

实际使用中，会优先选择**tanh**函数，原因是tanh函数相对于sigmoid函数来说梯度较大，收敛速度更快且引起梯度消失更慢。

* 1. 如何解决RNN中的梯度消失问题？

1.选取更好的激活函数，如Relu激活函数。ReLU函数的左侧导数为0，右侧导数恒为1，这就避免了“梯度消失“的发生。但恒为1的导数容易导致“梯度爆炸“，但设定合适的阈值可以解决这个问题。

2.加入BN层，其优点：加速收敛.控制过拟合，可以少用或不用Dropout和正则。降低网络对初始化权重不敏感，且能允许使用较大的学习率等。

3.改变传播结构，LSTM结构可以有效解决这个问题。

* 1. CNN VS RNN

文本

中度可信度描述已自动生成

不同点

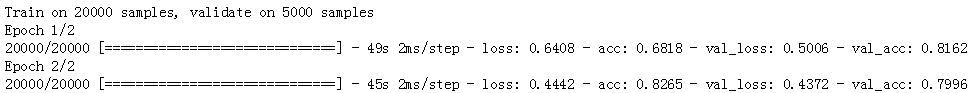
1.CNN空间扩展，神经元与特征卷积；RNN时间扩展，神经元与多个时间输出计算

2.RNN可以用于描述时间上连续状态的输出，有记忆功能，CNN用于静态输出。

* 1. Keras搭建RNN

文本

描述已自动生成



1. LSTM
   1. LSTM结构

**长短期记忆网络**（Long Short-Term Memory）是一种时间循环神经网络，是为了解决一般的RNN存在的长期依赖问题而专门设计出来的，所有的RNN都具有一种重复神经网络模块的链式形式。

三个门（**遗忘门，输入门，输出门**），两个状态（Ct,ht）

图示

描述已自动生成

遗忘门

卡通人物

描述已自动生成

作用对象：细胞状态 。

作用：将细胞状态中的信息选择性的遗忘。

Ft和Ct-1做点积操作，Ft确保Ct-1有哪些东西需要被遗忘调

输入层门

图示

描述已自动生成

作用对象：细胞状态

作用：将新的信息选择性的记录到细胞状态中。

操作步骤：

步骤一:sigmoid 层称 “输入门层” 决定什么值我们将要更新

步骤二，tanh 层创建一个新的候选值向量加入到状态中

输出层门

手机屏幕截图

描述已自动生成

作用对象：隐层ht 作用：确定输出什么值。

操作步骤：

步骤一:通过sigmoid 层来确定细胞状态的哪个部分将输出。

步骤二:把细胞状态通过 tanh 进行处理，并将它和 sigmoid 门的输出相乘，最终我们仅仅会输出我们确定输出的那部分。

* 1. LSTM结构推导，为什么比RNN好？

推导forget gate，input gate，cell state， hidden information等的变化；因为LSTM有进有出且当前的cell informaton是通过input gate控制之后叠加的，RNN是叠乘，因此LSTM可以防止梯度消失或者爆炸。

* 1. 为什么LSTM模型中既存在sigmoid又存在tanh两种激活函数，而不是选择统一一种sigmoid或者tanh？

sigmoid用在了各种**gate**上，产生0~1之间的值，一般只有sigmoid最直接了；

tanh用在了**状态**和**输出**上，是对数据的处理，这个用其他激活函数或许也可以。

* 1. LSTM中为什么经常是两层双向LSTM？

有些时候预测需要由前面若干输入和后面若干输入共同决定，这样会更加准确。

* 1. RNN扩展改进
     1. Bidirectional RNNs

将两层RNNs叠加在一起，当前时刻输出(第t步的输出)不仅仅与之前序列有关，还与之后序列有关。例如：为了预测一个语句中的缺失词语，就需要该词汇的上下文信息。Bidirectional RNNs是一个相对较简单的RNNs，是由两个RNNs上下叠加在一起组成的。输出由前向RNNs和后向RNNs共同决定。

* + 1. CNN-LSTMs

该模型中，CNN用于**提取对象特征**，LSTMs用于**预测**。CNN由于卷积特性，其能够快速而且准确地捕捉对象特征。LSTMs的优点：能够捕捉数据间的长时依赖性。

* + 1. Bidirectional LSTMs

有两层LSTMs。 一层处理过去的训练信息，另一层处理将来的训练信息。

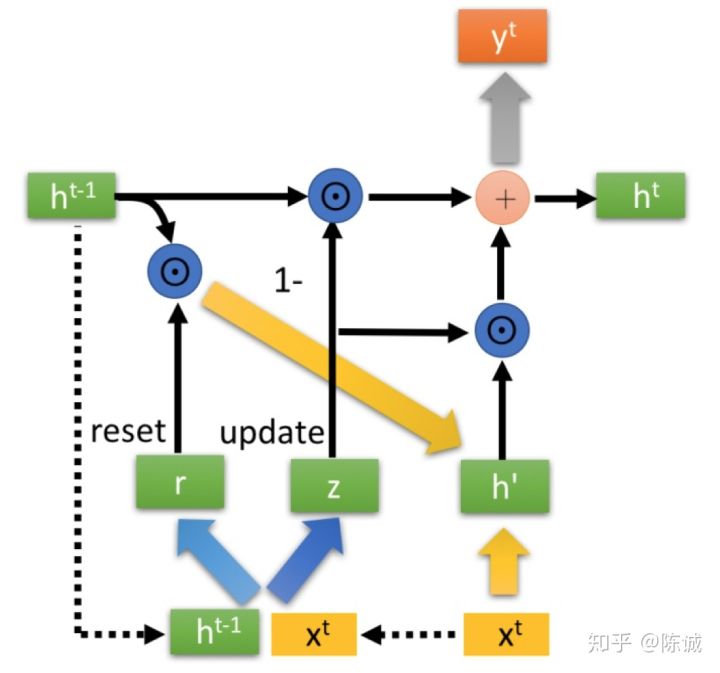
通过前向LSTMs获得前向隐藏状态，后向LSTMs获得后向隐藏状态，当前隐藏状态是前向隐藏状态与后向隐藏状态的组合。

* + 1. GRU

（14年提出）是一般的RNNs的变型版本，其主要是从以下两个方面进行改进。

1.以语句为例，序列中不同单词处的数据对当前隐藏层状态的影响不同，越前面的影响越小，即每个之前状态对当前的影响进行了距离加权，距离越远，权值越小。

2.在产生误差error时，其可能是由之前某一个或者几个单词共同造成，所以应当对对应的单词weight进行更新。GRUs的结构如下图所示。GRUs首先根据当前输入单词向量word vector以及前一个隐藏层状态hidden state计算出update gate和reset gate。再根据reset gate、当前word vector以及前一个hidden state计算新的记忆单元内容(new memory content)。当reset gate为1的时候，new memory content忽略之前所有memory content，最终的memory是由之前的hidden state与new memory content一起决定。



* 1. LSTM、RNN、GRU区别？

表格

描述已自动生成

与LSTM相比，GRU内部少了一个”门控“，参数比LSTM少，但是却也能够达到与LSTM相当的功能。考虑到硬件的计算能力和时间成本，因而很多时候我们也就会选择更加实用的GRU。

* 1. LSTM是如何实现长短期记忆功能的？
  2. LSTM的原理、写LSTM的公式、手推LSTM的梯度反向传播

1. 反向传播
   1. 什么是反向传播？

通俗解释：

类比几个人站成一排，第一个人看一幅画（输入数据），描述给第二个人（隐层）……依此类推，到最后一个人（输出）的时候，画出来的画肯定不能看了（误差较大）。

反向传播就是：把画拿给最后一个人看（**求取误差**），然后最后一个人就会告诉前面的人下次描述时需要注意哪里（**权值修正**）

一种与最优化方法（如梯度下降法）结合使用的，用来训练人工神经网络的常见方法。

目的是更新神经元参数,而神经元参数正是 z=wx+b 中的 (w,b).对参数的更新,利用损失值loss对参数的导数, 并沿着负梯度方向进行更新。

“正向传播”求损失，“反向传播”回传误差

* 1. 反向传播是如何工作的？

1.输入层接收x

2.使用权重w对输入进行建模

3.每个隐藏层计算输出，数据在输出层准备就绪

4.实际输出和期望输出之间的差异称为误差

5.返回隐藏层并**调整权重**，以便在以后的运行中减少此错误

这个过程一直重复，直到我们得到所需的输出。训练阶段在监督下完成。一旦模型稳定下来，就可以用于生产。

* 1. 为什么需要反向传播？

反向传播快速、简单且易于实现

没有要调整的参数

不需要网络的先验知识

模型不需要学习函数的特性

* 1. 手推BP

推导：**链式求导法**则反复用

文本

描述已自动生成

一些文字和图片的手机截图

描述已自动生成

一些文字和图片的手机截图

描述已自动生成

伪代码：

文本, 信件

描述已自动生成

1. GAN
   1. 什么是生成式对抗网络

**生成式对抗网络**,由一个生成器网络和一个**判别器**网络组成。判别器的训练目的是能够区分生成器的输出与来自训练集的真实图像，生成器的训练目的是欺骗判别器。值得注意的是，生成器从未直接见过训练集中的图像，它所知道的关于数据的信息都来自于判别器。

GAN 相关的技巧:

表格

描述已自动生成

* 1. 生成器

它将一个向量（来自潜在空间，训练过程中对其随机采样）转换为一张候选图像。GAN 常见的诸多问题之一，就是生成器“卡在”看似噪声的生成图像上。

一种可行的解决方案是在判别器和生成器中都使用 dropout。

文本

描述已自动生成

* 1. 判别器

接下来开发 discriminator 模型，它接收一张候选图像（真实的或合成的）作为输入，并将其划分到这两个类别之一：“生成图像”或“来自训练集的真实图像”。

文本

描述已自动生成

* 1. 训练技巧

输入规范化到（-1，1）之间，最后一层的激活函数使用tanh（BEGAN除外）

使用wassertein GAN的损失函数

如果有标签数据的话，尽量使用标签，也有人提出使用反转标签效果很好，另外使用标签平滑，单边标签平滑或者双边标签平滑

使用mini-batch norm， 如果不用batch norm 可以使用instance norm 或者weight norm

避免使用RELU和pooling层，减少稀疏梯度的可能性，可以使用leakrelu激活函数

优化器尽量选择ADAM，学习率不要设置太大，初始1e-4可以参考，另外可以随着训练进行不断缩小学习率

给D的网络层增加高斯噪声，相当于是一种正则

1. 超参数调整
   1. 神经网络中包含哪些超参数？

通常可以将超参数分为三类：网络参数、优化参数、正则化参数。

网络参数：可指网络层与层之间的交互方式（相加、相乘或者串接等）、卷积核数量和卷积核尺寸、网络层数（也称深度）和激活函数等。

优化参数：一般指学习率、批样本数量（batch size）、不同优化器的参数以及部分损失函数的可调参数。

正则化：权重衰减系数，丢弃比率（dropout）

* 1. 为什么要进行超参数调优？

本质上，这是模型优化寻找最优解和正则项之间的关系。网络模型优化调整的目的是为了寻找到全局最优解（或者相比更好的局部最优解），而正则项又希望模型尽量拟合到最优。两者通常情况下，存在一定的对立，但两者的目标是一致的，即最小化期望风险。模型优化希望最小化经验风险，而容易陷入过拟合，正则项用来约束模型复杂度。所以如何平衡两者之间的关系，得到最优或者较优的解就是超参数调整优化的目的。

* 1. 超参数的重要性顺序

首先，学习率，损失函数上的可调参数。在网络参数、优化参数、正则化参数中最重要的超参数可能就是学习率了。学习率直接控制着训综中网络梯度更新的量级，直接影响着模型的有效容限能力;损失函数上的可调参数，这些参数通常情况下需要结合实际的损失函数来调整，大部分情况下这些参数也能很直接的影响到模型的的有效容限能力。这些损失一般可分成三类，第一类辅助损失结合常见的损失函数，起到辅助优化特征表达的作用。例如度量学习中的Center loss，通常结合交叉粮损失伴随一个权重完成一些特定的任务。这种情况下一般建议辅助损失值不高于或者不低于交叉炉损失值的两个数量级;第二类，多任务模型的多个损失函数，每个损失函数之间或独立或相关，用于各自任务，这种情况取决于任务之间本身的相关性，目前笔者并没有一个普适的经验由于提供参考;第三类，独立损失函数，这类损失通常会在特定的任务有显著性的效果。例如RetinaNet中的focal oss，其中的参数γ、α对最终的效果会产生较大的影响，这类损失通常论文中会给出特定的建议值。

其次，批样本数量动量优化器的动量参数β，批样本决定了数量习度下降的方向。过小的批数量，极端情况下,例如batch size为1，即每个样本都去修正一次梯度方向，样本之间的差异越大越难以收敛。若网络中存在批归一化 (batchnorm) ，batch size过小则更难以收敛，甚至垮掉，这是因为数据样本越少，统计量越不具有代表性，噪声也相应的增加。而过大的batch size，会使得梯度方向基本稳定，容易陷入局部最优解，降低精度。一般参考范围会取在[1:1024]之间，当然这个不是绝对的，需要结合具体场景和样本情况，动量衰减参数β是计算梯度的指数加权平均数，并利用该值来更新参数，设置为0.9是一个常见且效果不错的选择;

最后，Adam优化器的超参数、权重衰减系数、丢弃比率(dropout) 和网络参数。在这里说明下，这些参数重要性放在最后并不等价于这些参数不重要。而是表示这些参数在大部分实践中不建议过多尝试，例如Adam优化器中的，常设为 0.9、0.999、10-8就会有不错的表现。权重衰减 系数通常会有个建议值，例如0.0005，使用建议值即可，不必过多兰试。dropout通常会在全连接层之间使用防止过拟合，建议比率控制在[0.2.0.5]之间。使用dropout时需要特别注意两点: 一、在RNN中，如果直接放在memory cell中,循环会放大噪声，扰乱学习。一般会建议放在输入和输出层; 二、不建议dropout后直接跟上batchnorm，dropout很可能影响batchnorm计算统计量，导至方差偏移，这种情况下会使得推理阶段出现模型完全垮掉的极端情况; 网络参数通常也属于超参数的范围内，通常情况下增加网络层数能增加模型的容限能力，但模型真正有效的容限能力还和样本数量和质量、层之间的关系等有关，所以一般情况下会选择先固定网络层数，调优到一定阶段或者有大量的硬件资源支持可以在网络深度上进行进一步调整.

* 1. 极端批样本数量下，如何训练网络？

极端批样本情况一般是指batch size为1或者batch size在6000以上的情况。这两种情况，在使用不合理的情况下都会导致模型最终性能无法达到最优甚至是崩溃的情况。

在目标检测、分割或者3D图像等输入图像尺寸较大的场景，通常batch size 会非常小。batch size过小通常对batchnorm的影响是最大的，若网络模型中存在batchnorm，batch size若只为1或者2时会对训练结果产生非常大的影响。这时通常有两种策略:

一、若模型使用了预训练网络，可冻结预训练网络中batchnorm的模型参数，有效降低batch size引起的统计量变化的影响。

二、在网络不是过深或者过于复杂时可直接移除batchnorm或者使用groupnorm代替batchnorm，前者不多阐释，后者是有FAIR提出的一种用于减少batch对batchnorm影响，其主要策略是先将特征在通道上进行分组，然后在组内进行归一化。即归一化操作上完全与batch size无关。这种groupnorm的策略被证实在极小批量网络训练上能达到较优秀的性能。当然这里也引入里group这个超参数，一般情况下建议不宜取group为1或者各通道单独为组的group数量，可结合实际网络稍加调试。

为了降低训练时间的成本，多机多卡的分布式系统通常会使用超大的batch size进行网络训练。可采用层自适应速率缩放（LARS）算法。从理论认知上将，batch size增大会减少反向传播的梯度更新次数，但为了达到相同的模型效果，需要增大学习率。但学习率一旦增大，又会引起模型的不收敛。为了解决这一矛盾，LARS算法就在各层上自适应的计算一个本地学习率用于更新本层的参数，这样能有效的提升训练的稳定性。

* 1. 什么是微调（fine-tune）

指稍微调整参数即可得到优秀的性能，是迁移学习的一种实现方式。微调和从头训练的本质区别在于模型参数的初始化，train from scratch通常指对网络各类参数进行随机初始化（当然随机初始化也存在一定技巧），随机初始化模型通常不具有任何预测能力，通常需要大量的数据或者特定域的数据进行从零开始的训练，这样需要训练到优秀的模型通常是稍困难的。而微调的网络，网络各类参数已经在其他数据集（例如ImageNet数据集）完成较好调整的，具备了较优秀的表达能力。因此，我们只需要以较小的学习速率在自己所需的数据集领域进行学习即可得到较为优秀的模型。微调通常情况下，无须再重新设计网络结构，预训练模型提供了优秀的结构，只需稍微修改部分层即可。在小数据集上，通常微调的效果比从头训练要好很多，原因在于数据量较小的前提下，训练更多参数容易导致过度拟合。

* 1. 微调有哪些不同方法？

以图像分类为例，通常情况下由于不同数据集需要的类别数不同，我们需要修改网络的输出顶层。这种情况下有两种微调方式：

不冻结网络模型的任何层，对最后的改动层使用较大的学习率，对未改动层以较小的学习率进行训练全模型训练，进行多轮训练即可。即一步完成训练。

冻结除了顶部改动层以外的所有层参数，即不对冻结部分的层进行参数训练更新，进行若干轮的微调训练后，放开顶部层以下的若干层或者全部放开所有层的参数，再次进行若干轮训练即可。即分多步训练。

以上两种都属于微调。目前由于存在大量优秀的预训练模型，如何确定哪个模型适合自己的任务并能得到最佳性能需要花大量的时间探索。此时，上述的前者是种不错训练方式，你无须进行过多分步的操作。而当探索到一个比较适合的模型时，你不妨可以再次重新尝试下以第二种方式进行训练，或许能得到相比于前者稍高些的性能，因为小数据集上调整过多的参数过拟合的机率也会增大，当然这并不是绝对的。

* 1. 微调先冻结底层，训练顶层的原因？

首先冻结除了顶部改动层以外的所有层参数，对顶层进行训练，这个过程可以理解为顶层的域适应训练，主要用来训练适应模型的现有特征空间，防止顶层糟糕的初始化，对已经具备一定表达能力的层的干扰和破坏，影响最终的性能。之后，在很多深度学习框架教程中会使用放开顶层往下一半的层数，继续进行微调。这样的好处在于越底层的特征通常是越通用的特征，越往上其整体的高层次语义越完备，这通过感受野很容易理解。所以，若预训练模型的数据和微调训练的数据语义差异越大（例如ImageNet的预模型用于医学图像的训练），那越往顶层的特征语义差异就越大，因此通常也需要进行相应的调整。

* 1. 不同的数据集特性下如何微调？

数据集数据量少，数据和原数据集类似。这是通常做法只需修改最后的输出层，训练即可，训练过多参数容易过拟合。

数据集数据量少，数据和原数据集差异较大。由于数据差异较大，可以在完成输出顶层的微调后，微调顶层往下一半的层数，进行微调。

数据集数据量大，数据与原数据集差异较大。这种情况下，通常已经不需要用预训练模型进行微调，通常直接重新训练即可。

数据集数据量大，数据与原数据类似。这时预训练模型的参数是个很好的初始化，可利用预训练模型放开所有层以较小的学习率微调即可。

* 1. 目标检测中使用预训练模型的优劣？

目标检测中无论是一阶段的YOLO、SSD或者RetinaNet 还是二阶段的Faster R-CNN、R-FCN 和 FPN都是基于ImageNet上预训练好的分类模型。

优势在于：

1、正如大部分微调的情况一样，使用预训练网络已拥有优秀的语义特征，能有效的加快训练速度；

2、其次，对于大部分二阶段的模型来说，并未实现严格意义上的完全端对端的训练，所以使用预训练模型能直接提取到语义特征，能使两个阶段的网络更容易实现模型的优化。

劣势在于，分类模型和检测模型之间仍然存在一定任务上的差异：

1、分类模型大部分训练于单目标数据，对同时进行多目标的捕捉能力稍弱，且不关注目标的位置，在一定程度上让模型损失部分空间信息，这对检测模型通常是不利的；

2、域适应问题，若预训练模型（ImageNet）和实际检测器的使用场景（医学图像，卫星图像）差异较大时，性能会受到影响；

3、使用预训练模型就意味着难以自由改变网络结构和参数限制了应用场合。

* 1. 目标检测中如何从零开始训练(train from scratch)？

1、数据集不大时，同样需要进行数据集增强。

2、预训练模型拥有更好的初始化，train from scratch需要更多的迭代次数以及时间训练和优化检测器。而二阶段模型由于并不是严格的端对端训练，此时可能需要更多的迭代次数以及时间，而一阶段检测模型训练会相对更容易些。

3、目标检测中train from scratch最大的问题还是batch size过小。所以可采取的策略是增加GPU使用异步batchnorm增大batch size，若条件限制无法使用更多GPU时，可使用groupnorm代替batchnorm

4、由于分类模型存在对多目标的捕捉能力弱以及对物体空间位置信息不敏感等问题，可借鉴DetNet训练一个专属于目标检测的模型网络，增强对多目标、尺度和位置拥有更强的适应性。

* 1. 自动化超参数搜索方法有哪些？

主要包含网格搜索，随机搜索，基于模型的超参优化

7.6.1网格搜索：

通常当超参数量较少的时候，可以使用网格搜索法。即列出每个超参数的大致候选集合。利用这些集合 进行逐项组合优化。在条件允许的情况下，重复进行网格搜索会当优秀，当然每次重复需要根据上一步得到的最优参数组合，进行进一步的细粒度的调整。网格搜索最大的问题就在于计算时间会随着超参数的数量指数级的增长。

7.6.2随机搜索：

随机搜索，是一种用来替代网格搜索的搜索方式。随机搜索有别于网格搜索的一点在于，不需要设定一个离散的超参数集合，而是对每个超参数定义一个分布函数来生成随机超参数。随机搜索相比于网格搜索在一些不敏感超参上拥有明显优势。例如网格搜索对于批样本数量（batch size），在[16,32,64]这些范围内进行逐项调试，这样的调试显然收益更低下。当然随机搜索也可以进行细粒度范围内的重复的搜索优化。

7.6.3基于模型的超参优化：

调优问题转化为了优化问题。直觉上会考虑是否进行一个可导建模，然后利用梯度下降进行优化。但不幸的是我们的超参数通常情况下是离散的，而且其计算代价依旧很高。

基于模型的搜索算法，最常见的就是贝叶斯超参优化。有别于的网格搜索和随机搜索独立于前几次搜索结果的搜索，贝叶斯则是利用历史的搜索结果进行优化搜索。其主要有四部分组成，

目标函数，大部分情况下就是模型验证集上的损失。

搜索空间，即各类待搜索的超参数。

优化策略，建立的概率模型和选择超参数的方式。

4、历史的搜索结果。首先对搜索空间进行一个先验性的假设猜想，即假设一种选择超参的方式，然后不断的优化更新概率模型，最终的目标是找到验证集上误差最小的一组超参数。

* 1. 为什么在模型训练开始会有warmup？

warmup,在刚刚开始训练时以很小的学习率进行训练，使得网络熟悉数据，随着训练的进行学习率慢慢变大，到了一定程度，以设置的初始学习率进行训练，接着过了一些inter后，学习率再慢慢变小；学习率变化：上升——平稳——下降。

有助于减缓模型在初始阶段对mini-batch的提前过拟合现象，保持分布的平稳；

有助于保持模型深层的稳定性

1. 损失函数大全
   1. L1范数损失 L1Loss

计算output和target之差的绝对值

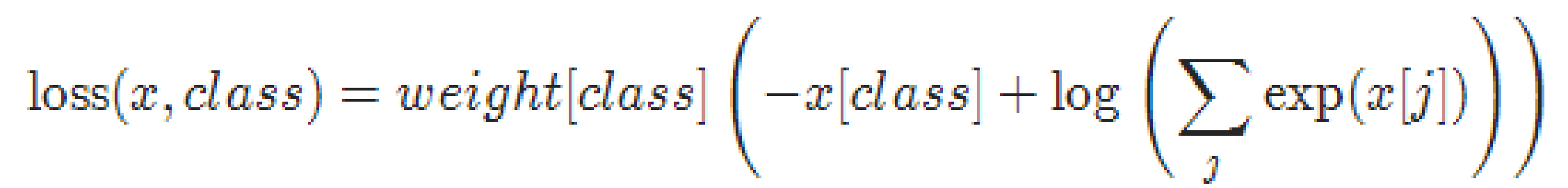
* 1. 均方误差损失 MSELoss

计算output和target之差的均方误差

* 1. 交叉熵损失 CrossEntropyLoss

当训练有C个类别的分类问题时很有效.可选参数weight必须是一个1维Tensor,权重将被分配给各个类别.对于不平衡的训练集非常有效。

在多分类任务中，经常采用softmax激活函数+交叉熵损失函数，因为交叉熵描述了两个概率分布的差异，然而神经网络输出的是向量，并不是概率分布的形式。所以需要softmax激活函数将一个向量进行“归一化”成概率分布的形式，再采用交叉熵损失函数计算loss。



* 1. KL散度损失 KLDivLoss

计算input和target之间的KL散度。KL散度可用于衡量不同的连续分布之间的距离,在连续的输出分布的空间上(离散采样)上进行直接回归时很有效

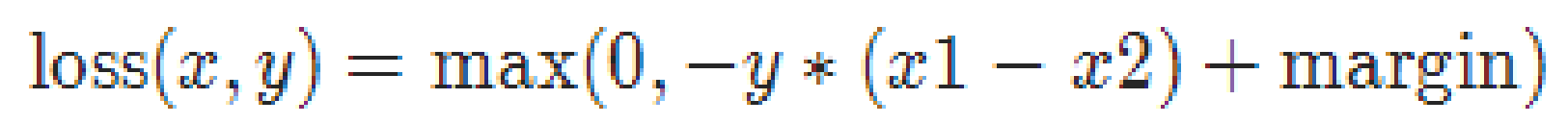
* 1. 二进制交叉熵损失 BCELoss

二分类任务时的交叉熵计算函数。用于测量重构的误差,例如自动编码机，注意目标的值t[i]的范围为0到1之间

* 1. BCEWithLogitsLoss

BCEWithLogitsLoss损失函数把Sigmoid层集成到了BCELoss类中该版比用一个简单的Sigmoid层和BCELoss在数值上更稳定,因为把这两个操作合并为一个层之后,可以利用log-sum-exp的技巧来实现数值稳定

* 1. MarginRankingLoss



* 1. HingeEmbeddingLoss

图片包含 图表

描述已自动生成

* 1. 多标签分类损失 MultiLabelMarginLoss

图形用户界面, 文本

中度可信度描述已自动生成

* 1. 平滑L1损失 SmoothL1Loss

图片包含 文本

描述已自动生成

手机屏幕的截图

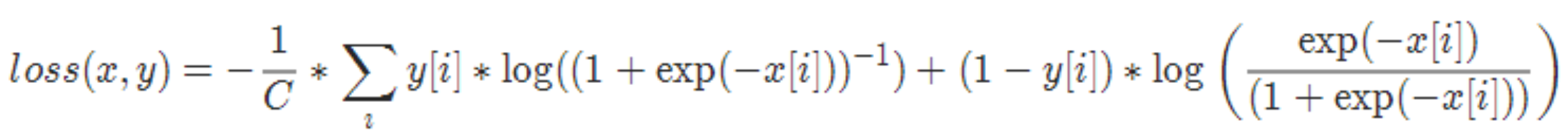
描述已自动生成

* 1. 分类的Logistic损失 SoftMarginLoss

图片包含 图形用户界面

描述已自动生成

* 1. 多标签one-versus-all损失 MultiLabelSoftMarginLoss

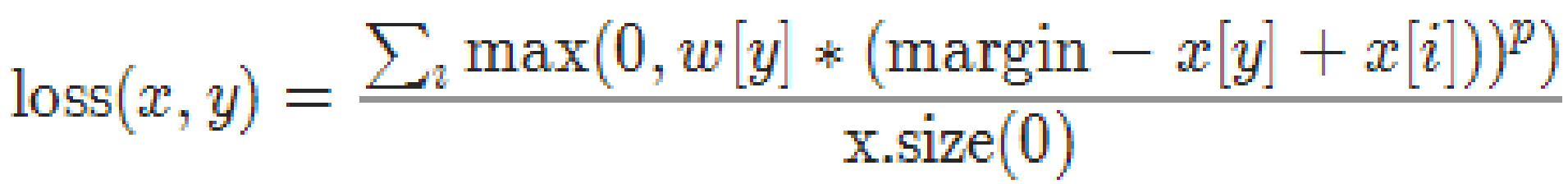


* 1. cosine损失 CosineEmbeddingLoss

文本

描述已自动生成

* 1. 多类别分类的hinge损失 MultiMarginLoss



* 1. 三元组损失 TripletMarginLoss

给一个A，然后再给B、C，看看B、C谁和A更像

* 1. 连接时序分类损失 CTCLoss

CTC连接时序分类损失，可以对没有对齐的数据进行自动对齐，主要用在没有事先对齐的序列化数据训练上。比如语音识别、ocr识别等等

* 1. 负对数似然损失 NLLLoss

负对数似然损失.用于训练C个类别的分类问题

* 1. NLLLoss2d

对于图片输入的负对数似然损失.它计算每个像素的负对数似然损失