计算机保研面试_机器学习常见题

计算机保研面试_机器学习常见题
1 机器学习
(1) 常见的机器学习算法
(2) 回归和分类
(3) k-means
(4) PCA
(5) KNN
(6) 决策树
(7) 朴素贝叶斯
(8) 最小二乘法
(9) SVM
(10) 逻辑回归
2 深度学习
(1) 梯度下降, BGD, SGD, Mini-batch SGD, Momentum
(2) 自适应优化器, Adagrad, RMSprop, Adam
(3) 梯度下降陷入局部最优怎么办
(4) Batch Normalization
(5) 梯度爆炸和梯度消失
(6) 空洞卷积
(7) 判别式和生成式模型
(8) VIT
(9) 深度可分离卷积
(10) Transformer为什么需要进行Multi-head Attention?
(11) 如何确定CNN卷积核通道数和卷积输出层的通道数
(12) CNN的特点以及优势
(13) 池化层作用
(14) RNN
(15) 有哪些办法防止过拟合
(16) 简述梯度下降法和牛顿法的优缺点？梯度下降法和牛顿法区别
(17) L1和L2正则化
(18) Batch Normalization 和 Group Normalization有何区别?
(19) dropout的原理，为什么可以防止过拟合*
(20) 简述一下数据增强的方法
(21) 1x1大小的卷积核的作用
(22) 感受野的计算
(23) GAP
(24) Dropout层有什么作用
(25) 为什么能缓解过拟合
(26) sigmoid和relu激活函数、有什么区别
(27) relu函数为什么能使网络稀疏等等
3 PyTorch
(1) Conv2d
(2) 训练一个resnet
(3) 怎么看loss和acc的变化
(4) MaxPooling
(5) 卷积、BN、激活函数的顺序
(6) BN
(7) dropout
(8) BN在训练和测试时的差别

(9) Dropout 在训练和测试时都需要嘛?

1 机器学习

(1) 常见的机器学习算法

- 监督学习：决策树、朴素贝叶斯、最小二乘法、逻辑回归、SVM、集成学习等。
- 无监督学习：k-means、PCA、SVD 分解、GMM 等。

(2) 回归和分类

回归算法是一种有监督学习算法，用来建立自变量 X 和观测变量 Y 之间的映射关系，如果观测变量是**离散**的，则称其为**分类**；如果观测变量是**连续**的，则称其为**回归。**

(3) k-means

- 算法过程：
    1. 选择初始的 k 个样本作为初始聚类中心
    2. 对于每个样本，计算它到 k 个聚类中心的距离，并将其分到距离最小的聚类中心所在的那一类
    3. 对于每个类别，重新计算它的聚类中心
    4. 不断重复 2, 3 ，直到达到终止条件
- 缺点：
    1. k 值需要人为设定。
    2. 对初始聚类中心敏感，对异常值敏感，需要对数据进行预处理（归一化、标准化）。
    3. 收敛速度慢，一种改进方法：第一次迭代时，计算每个点到所有 k 个质心的距离，后续迭代中，仅计算上一次迭代中离这个节点最近的某几个质心的距离。

(4) PCA

主成分分析法是一种无监督学习方法，常用于数据降维，它的目标是通过某种线性投影，将高维的数据映射到低维的空间中表示，并期望在所投影的维度上数据的方差最大，以此使用较少的数据维度，同时保留住较多的原数据点的特性。

输入：样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
低维空间维数 d'.
过程：
1: 对所有样本进行中心化: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i$;
2: 计算样本的协方差矩阵 $\mathbf{X}\mathbf{X}^{\mathrm{T}}$;
3: 对协方差矩阵 $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ 做特征值分解;
4: 取最大的 d' 个特征值所对应的特征向量 $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'}$.
输出：投影矩阵 $\mathbf{W}^* = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'})$.

(5) KNN

- 一种监督学习方法。
- 对于待判断的点，基于某种距离度量，找到离它最近的几个数据点，根据它们的类型决定待判断点的类型。
    - 在**分类**任务中可使用“投票法”，即选择这 k 个样本中出现最多的类别标记作为预测结果；
    - 在**回归**任务中可使用“平均法”，即将这 k 个样本的实值输出标记的平均值作为预测结果。

(6) 决策树

- 决策树是一颗由多个判断节点组成的树。在使用模型进行预测时，根据输入参数依次在各个判断节点进行判断，最后到叶子节点即为预测结果。
- 构造决策树的主要步骤
    1. 遍历每个决策条件，对结果集进行拆分。
    2. 计算在该决策条件下所有可能的拆分情况的信息增益，信息增益最大的拆分为本次最优拆分。
    3. 重复 1, 2 两步，直至信息增益<=0。
- 剪枝：防止过拟合
    - 预剪枝：在决策树生成过程中，对每个结点在划分前先进行估计，若当前结点的划分不能带来决策树泛化性能提升，则停止划分并将当前结点标记为叶结点。
    - 后剪枝：先从训练集生成一棵完整的决策树，然后自底向上地对非叶节点进行考察，若将该结点对应的子树替换为叶节点能带来决策树泛化性能提升，则将该子树替换为叶结点。
- 选择划分训练数据集的特征
    - ID3 算法：选择信息增益最大的特征。（存在偏向于选择取值较多的特征的问题）
    - C4.5 算法：选择信息增益比最大的特征。
- 缺点：未考虑属性间依赖、容易过拟合、不可用于推测属性缺失的样本。

(7) 朴素贝叶斯

- 朴素贝叶斯：一种基于贝叶斯定理与特征条件独立假设的分类方法。

$$P(\text{类别}|\text{特征}) = \frac{P(\text{特征}|\text{类别})P(\text{类别})}{P(\text{特征})} = \frac{P(\text{特征}|\text{类别})P(\text{类别})}{\sum_i P(\text{特征}|\text{类别}_i)P(\text{类别}_i)}$$

- 朴素贝叶斯朴素在哪里
    - 各特征之间相互独立。
    - 每个特征同等重要。

(8) 最小二乘法

最小二乘法：通过最小化误差的平方和，使得拟合对象最大限度逼近目标对象。如在线性回归中，最小二乘法就是试图找到一条直线，使所有样本到直线上的欧氏距离之和最小。

$$E=\sum_{i=1}^{n}e_i^2=\sum_{i=1}^{n}(y_i-\hat{y})^2$$

(9) SVM

- 基本思想：在二分类问题上，SVM 希望找到一个超平面，使得两类数据离超平面越远越好，从而对新的数据分类更准确，即使分类器更加鲁棒，其学习策略就是不断最大化支撑向量到分隔超平面的距离。

  > 支持向量：离分隔超平面最近的那些点。

- 核技巧（Kernel Trick）：有时候数据线性不可分，那么可以用核函数将特征从低维映射到高维进行转换，但是它是先在低维上进行计算，实际的分类效果表现在高维上。（用核函数替代向高维空间的非线性映射）
- 优点
  1. 适用于小样本的机器学习任务。
  2. 可解释性强。
  3. 可以通过核函数来解决非线性问题
- 缺点
  1. 不能有效处理大规模样本，因为 SVM 是借助二次规划来求解支持向量，而求解二次规划将涉及 N 阶矩阵的计算（N为样本的个数），当 N 数目很大时该矩阵的存储和计算将耗费大量的机器内存和运算时间。
  2. 用 SVM 解决多分类问题存在困难。

(10) 逻辑回归

逻辑回归=线性回归+sigmoid函数

2 深度学习

[深度学习面试题]

[深度学习-算法工程师岗位面试常见问题及解答]

(1) 梯度下降, BGD, SGD, Mini-batch SGD, Momentum

- 梯度下降基本原理：损失函数沿着梯度向量相反的方向减小得最快，更容易到达最小值。
- 批梯度下降（BGD）：最原始的，每次使用所有的样本来计算梯度并求平均，作为该次迭代的梯度。

$$\vec{x}\leftarrow\vec{x}-\eta\nabla f(\vec{x})$$

$$\nabla f(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\vec{x})$$

    - 优点：
        - 一次迭代是对所有样本进行计算，此时利用矩阵进行操作，实现了**并行。**
        - 当目标函数为凸函数时，BGD一定能够得到**全局最优。**
    - 缺点：收敛慢。

- 随机梯度下降（SGD）：每次随机抽取某一个样本来计算梯度，作为该次迭代的梯度。

$$\vec{x} \leftarrow \vec{x} - \eta\nabla f_i(\vec{x})$$

    - 优点：训练速度快。
    - 缺点：局部最优、不利于并行。

- 小批量随机梯度下降（Mini-batch SGD）：每次使用一部分（batch）的样本来计算梯度，作为该次迭代的梯度。

$$\vec{x}_t \leftarrow \vec{x}_{t-1} - \eta_t \nabla f_{\Psi_t}(\vec{x}_t)$$
$$\nabla f_{\Psi_t}(\vec{x}_t) = \frac{1}{B}\sum_{i\in\Psi_t}\nabla f_i(\vec{x}_{t-1})$$

    - 优点：结合前两者的优点。
    - 缺点：学习率不好调，太小了收敛慢，太大了损失函数会在极小值处不停地震荡甚至偏离。
    - 注意：一般选取的 batch size 越大，训练的 epoch 也要增加。

- 动量随机梯度下降（Momentum SGD）：采用指数加权移动平均，每次的更新方向由当前的负梯度方向和上一次的迭代更新方向加权组合形成。（如果梯度方向不变，就越发更新的快，反之减弱当前梯度）

$$v_t \leftarrow \gamma v_{t-1} + \eta_t \nabla f_t; where, \gamma \in [0,1]$$
$$\vec{x}_t \leftarrow \vec{x}_{t-1} - v_t$$

    - 优点：加快收敛，抑制上下震荡，有一定摆脱局部最优的能力。

(2) 自适应优化器, Adagrad, RMSprop, Adam

- Adagrad：对于每个参数，使用一个自适应的学习率，开始很大，随着训练的进行，慢慢变小。具体来说，就是先初始化一个变量 s 为 0 ，之后每次将该参数的梯度平方和累加到 s 上，在更新该参数时，学习率除以 $\sqrt{s}$ 。
    - 优点：自适应的调节学习率，加快收敛。
    - 缺点：
        - 学习率下降得太快。
        - 在训练的中后期，分母上梯度平方的累加将会越来越大，从而梯度趋近于 0，使得训练提前结束。
- RMSprop：为了解决 Adagrad 学习率急剧下降问题而提出的，它在更新变量 s 时，不是直接将该参数的梯度平方和累加到 s 上，而是做了指数加权移动平均。

$$s_t = \gamma s_{t-1} + (1-\gamma) g_t \odot g_t$$

- Adam：相当于 RMSprop + Momentum，Adam可能不收敛。

(3) 梯度下降陷入局部最优怎么办

可以用 BGD、SGD、Mini-batch SGD、Momentum、RMSprop、Adam 等方法来避免陷入局部最优。

(4) Batch Normalization

BN：缓解梯度消失、加快收敛（BN后抑制了损失函数不平滑）、防止过拟合（引入噪声）

BN就是为了让网络在每一层都保持相近的分布

BN层为什么能够加快网络收敛速度

- BN就是通过一定的规范化手段，把每层神经网络任意神经元这个输入值的分布强行拉回到均值为0方差为1的标准正态分布，其实就是把越来越偏的分布强制拉回比较标准的分布
- 这样使得激活输入值落在非线性函数对输入比较敏感的区域，这样输入的小变化就会导致损失函数较大的变化，意思是这样让梯度变大，避免梯度消失问题产生，而且梯度变大意味着学习收敛速度快，能大大加快训练速度。

BN层为什么能够避免梯度消失

见上一问

为什么BN层一般用在线性层和卷积层后，而不是放在非线性单元后

- 因为非线性单元的输出分布形状会在训练过程中变化，归一化无法消除他的方差偏移。
- 相反的，全连接和卷积层的输出一般是一个对称,非稀疏的一个分布，更加类似高斯分布，对他们进行归一化会产生更加稳定的分布。
- 其实想想也是的，像relu这样的激活函数，如果你输入的数据是一个高斯分布，经过他变换出来的数据能是一个什么形状？小于0的被抑制了，也就是分布小于0的部分直接变成0了，这样不是很高斯了。
- **BN计算均值的时候计算多少次**：假设[b,c,h,w],计算b*h*w次，因为是相当于对每个channel计算
- **BN共有多少个参数**：2*c个参数

BN:(batchnorm)是在batch上，对NHW做归一化，对小batchsize效果不好
LN:layernorm在通道方向上，对CHW做归一化，主要对RNN作用明显
IN:instancenorm在图像像素上，对HW做归一化，用在风格迁移上
GN:(groupnorm)将channel分组，然后再做归一化
switchableNorm将BN,LN,IN结合，赋予权重，让网络自己去学习归一化层应该使用什么方法。

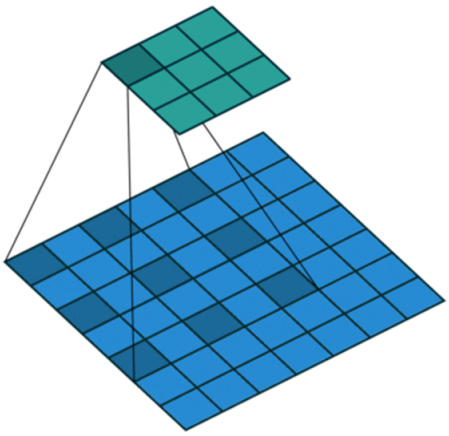
(5) 梯度爆炸和梯度消失

- 定义：**反向传播**过程中需要**对激活函数求导**，若导数大于 1，那么随着网络层数的增加梯度更新将会朝着指数爆炸的方式增加，这就是梯度爆炸；若导数小于 1，那么随着网络层数的增加梯度更新信息会朝着指数衰减的方式减少，这就是梯度消失。

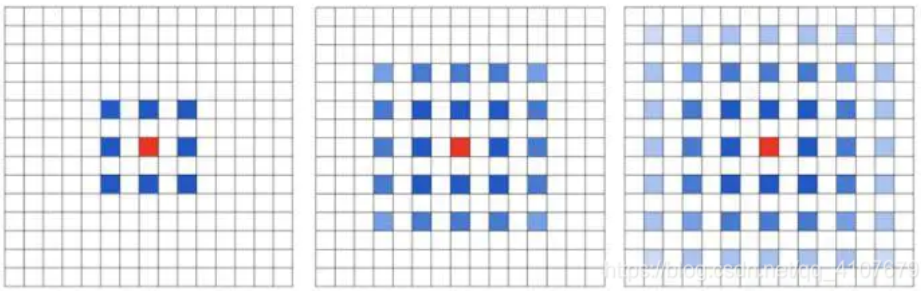
- 解决方法：
    1. 使用 ReLU 激活函数，因为他的导数为 1，缓解梯度爆炸和梯度消失。
    2. 梯度截断，梯度超过阈值就截断，缓解梯度爆炸。
    3. 添加权重正则项，权重越大，则惩罚越大，常见的有 L1 正则项（权重绝对值）和 L2 正则项（权重平方），缓解梯度爆炸。
    4. Batch Normalization，缓解梯度爆炸和梯度消失。
    5. 残差连接，输出值加上输入值再传到下一层网络，缓解梯度消失。
    6. 使用 LSTM 等自循环和门控机制，缓解梯度消失。

(6) 空洞卷积

- Atrous Convolution：在标准的卷积 map 里注入空洞



- 优点：
    - 扩大感受野：既不增加参数量，也不会像池化那样损失信息
    - 捕获多尺度上下文信息：通过改变 dilation rate，可以改变感受野，就可以捕获多尺度信息
- 问题：
    - 网格效应：多次堆叠具有相同 dilation rate 的空洞卷积会导致一些像素始终不参与运算，分布呈网格，这对于像素级别的预测是不友好的

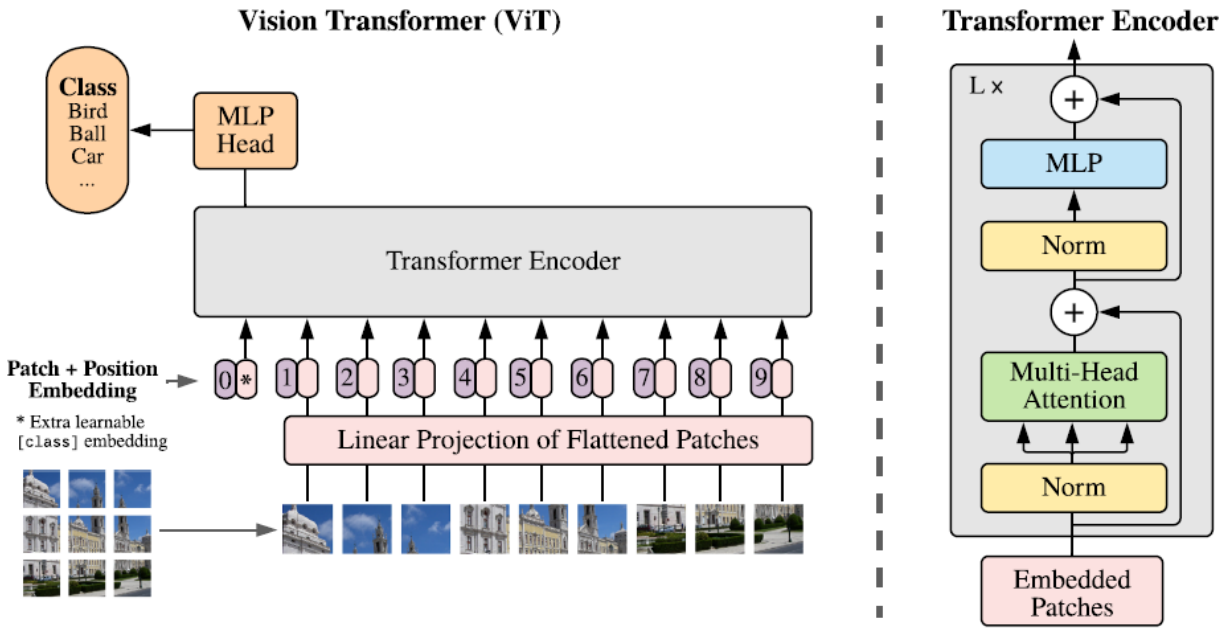


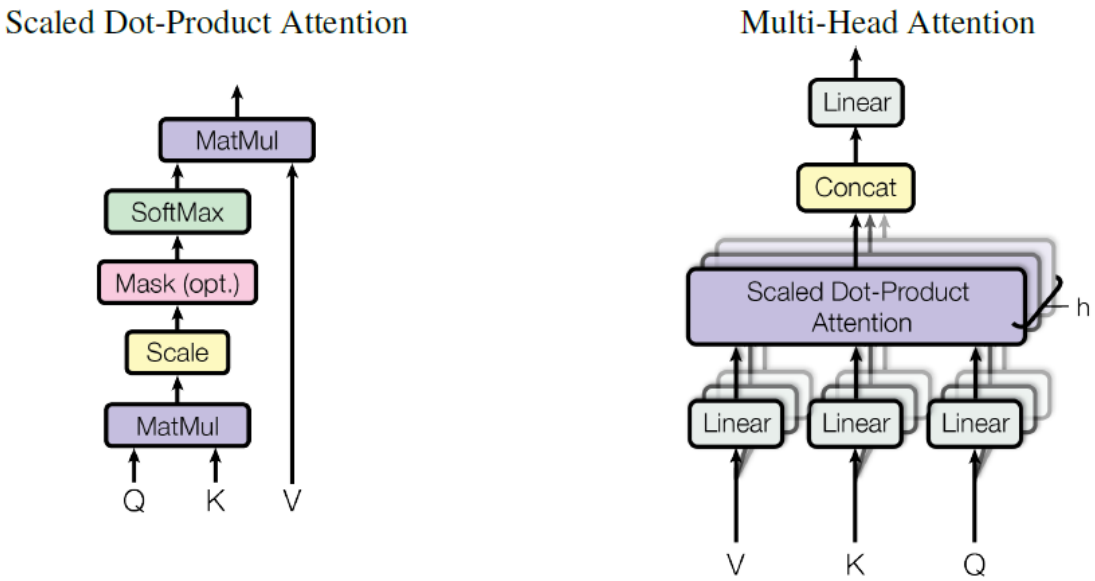
    - 对于一些小物体，它不需要很大的感受野，空洞就多余了
    - 解决方案：用多个不同 dilation rate 的空洞卷积核混合，dilation rate 需要遵循一定的规则

(7) 判别式和生成式模型

- 监督学习分为判别式和生成式
  - 判别式：对条件概率 P(Y|X) 建模。如：感知机、决策树、KNN、神经网络、SVM
  - 生成式：对联合概率 P(X, Y) 建模。如：贝叶斯、GMM、HMM、VAE、GAN
- 举个栗子：判断一个水果是苹果还是梨子
  - 判别式：从历史数据中学习到 P(Y|X)，由水果 X 直接得出 P(苹果|X) 和 P(梨子|X)
  - 生成式：从历史数据中学习到苹果模型和梨子模型，再将水果 X 输入两个模型看看哪个概率大
- 优缺点：生成式模型收敛速度更快，可以应付隐变量的情况，但计算资源较大；判别式反之
- ‘关系：由生成模型可以得到判别模型，但由判别模型得不到生成模型

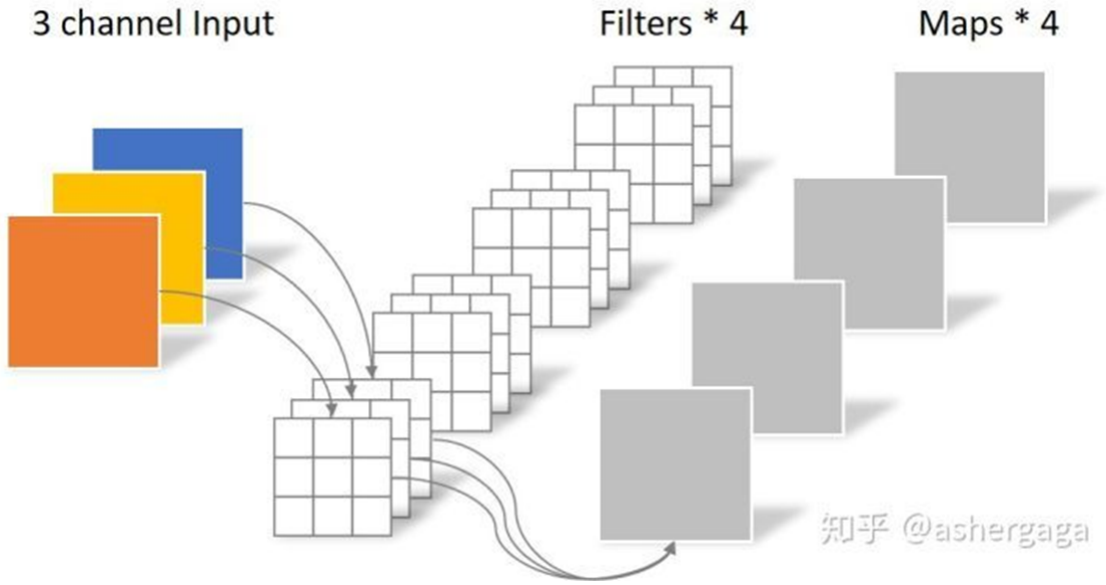
(8) VIT

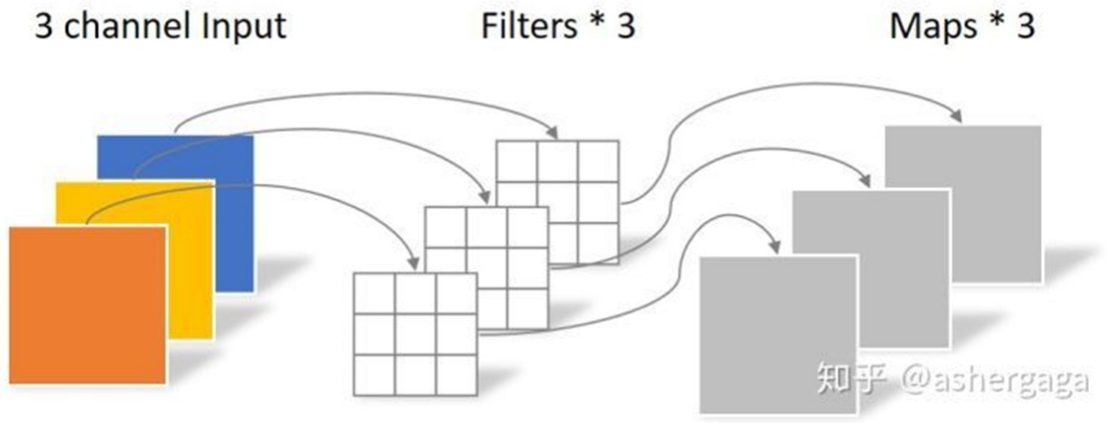


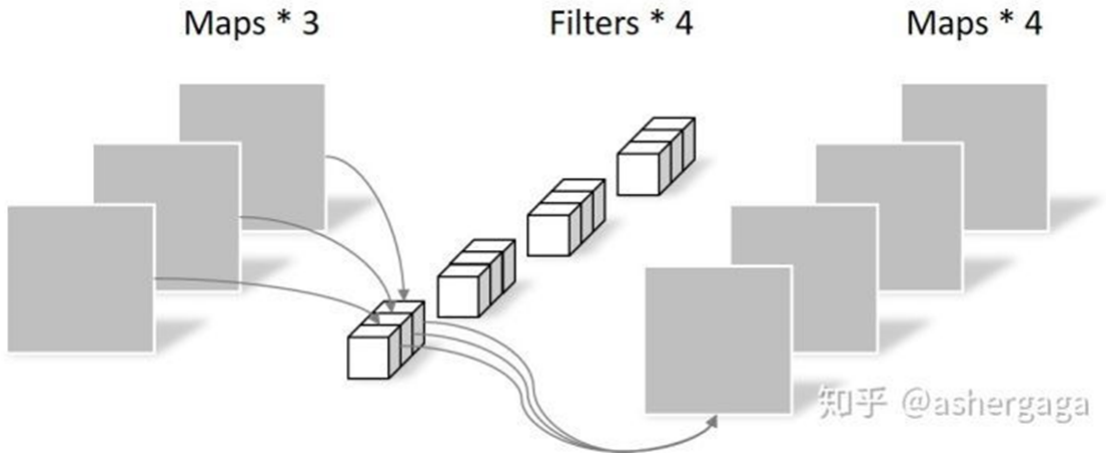


(9) 深度可分离卷积

受到MobileNet的启发，深度可分离卷积 =逐通道卷积 + 逐点卷积，可以减少参数量







(10) Transformer为什么需要进行Multi-head Attention?

可以让模型去关注不同方面的信息，最后再将各个方面的信息综合起来, 有助于网络捕捉到更丰富的特征或者信息

(11) 如何确定CNN卷积核通道数和卷积输出层的通道数

CNN的卷积核通道数 = 卷积输入层的通道数；CNN的卷积输出层通道数 = 卷积核的个数

(12) CNN的特点以及优势

CNN使用范围是具有局部空间相关性的数据，比如图像，自然语言，语音

局部连接：可以提取局部特征。
权值共享：减少参数数量，因此降低训练难度（空间、时间消耗都少了）。可以完全共享，也可以局部共享（比如对人脸，眼睛鼻子嘴由于位置和样式相对固定，可以用和脸部不一样的卷积核）

降维：通过池化或卷积stride实现。
多层次结构：将低层次的局部特征组合成为较高层次的特征。不同层级的特征可以对应不同任务。

(13) 池化层作用

保留主要的特征同时减少参数和计算量，防止过拟合，提高模型泛化能力

(14) RNN

循环神经网路，即一个序列当前的输出与前面的输出也有关

(15) 有哪些办法防止过拟合

1. Dropout
2. 加L1/L2正则化
3. Batch Normalization
4. 网络bagging
5. 提取终止训练
6. 数据增强

(16) 简述梯度下降法和牛顿法的优缺点？梯度下降法和牛顿法区别

1.牛顿法：是通过求解目标函数的一阶导数为0时的参数，进而求出目标函数最小值时的参数。①收敛速度很快。②海森矩阵的逆在迭代过程中不断减小，可以起到逐步减小步长的效果。③缺点：海森矩阵的逆计算复杂，代价比较大，因此有了拟牛顿法。

2.梯度下降法：是通过梯度方向和步长，直接求解目标函数的最小值时的参数。越接近最优值时，步长应该不断减小，否则会在最优值附近来回震荡。

(17) L1和L2正则化

L1正则化和L2正则化的区别

L1与L2正则的联系与区别

(18) Batch Normalization 和 Group Normalization有何区别?

Batch Normalization就是在batch纬度上做正交归一化，GN就是在channel维度上对每个group做正交归一化

(19) dropout的原理，为什么可以防止过拟合*

dropout的原理就是在网络前向传播的时候，让神经元的激活值以一定的概率变为零，这样可以使模型的泛化性能更强。

前向：训练时，利用伯努利分布，随机选出一个只包含0,1的mask矩阵，然后用这个mask矩阵去对应乘上每个输入得到的就是Dropout后的结果，再除以（1-p）；测试的时候不用Dropout
反向：训练时根据mask来求对应的梯度，测试时无Dropout

dropout为什么可以防止过拟合呢?

1、dropout其实相当于我们日常用到的基于平均的ensemble，ensemble有两种方式，基于平均的ensemble和投票的ensemble。对于网络中的部分神经元进行概率暂时舍弃，这样相当于训练了多个网络。
2、dropout还取消了神经元之间的共适应关系，使得网络的输出不依赖于网络中的某些隐含节点的固定作用，使模型的鲁棒性更好。
3、类似于生物进化的角色，环境的变化不会对物种造成毁灭性的影响。

(20) 简述一下数据增强的方法

主要分为离线增强和在线增强的方法。
离线增强是指数据集在本地进行处理。
在线增强：翻转（水平，垂直），旋转，缩放，裁剪，平移，添加噪声等。

(21) 1x1大小的卷积核的作用

通过控制卷积核个数实现升维或者降维，从而减少模型参数
对不同特征进行归一化操作（BN）,增加非线性（relu）
用于不同channel上特征的融合

(22) 感受野的计算

No.	Layers	Kernel Size	Stride
1	Conv1	3*3	1
2	Pool1	2*2	2
3	Conv2	3*3	1
4	Pool2	2*2	2
5	Conv3	3*3	1
6	Conv4	3*3	1
7	Pool3	2*2	2

感受野初始值 $l_0 = 1$，每层的感受野计算过程如下：

$l_0 = 1$
$l_1 = 1 + (3-1) = 3$
$l_2 = 3 + (2-1) * 1 = 4$
$l_3 = 4 + (3-1) * 1 * 2 = 8$
$l_4 = 8 + (2-1) * 1 * 2 * 1 = 10$
$l_5 = 10 + (3-1) * 1 * 2 * 1 * 2 = 18$
$l_6 = 18 + (3-1) * 1 * 2 * 1 * 2 * 1 = 26$
$l_7 = 26 + (2-1) * 1 * 2 * 1 * 2 * 1 * 1 = 30$

(23) GAP

优点：

1. 和全连接层相比，使用全局平均池化技术，对于建立特征图和类别之间的关系，是一种更朴素的卷积结构选择。
2. 全局平均池化层不需要参数，避免在该层产生过拟合。
3. 全局平均池化对空间信息进行求和，对输入的空间变化的鲁棒性更强

缺点：

1. 收敛速度变慢

(24) Dropout层有什么作用

(25) 为什么能缓解过拟合

(26) sigmoid和relu激活函数、有什么区别

(27) relu函数为什么能使网络稀疏等等

3 PyTorch

PyTorch面试的50个问题 - 芒果文档 (imangodoc.com)

(119条消息) CV面试题（持续更新！！！）*just-solo的博客-CSDN博客*pytorch面试题

(1) Conv2d

class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)

(2) 训练一个resnet

```
net = torchvision.models.resnet101()
epochs = 1000
lr = 0.001
loss_fun = torch.nn.CrossEntropyLoss()
opt_SGD = torch.optim.SGD(net.parameters(), lr=lr)
data = data.cuda()
label = label.cuda()
for epoch in range(epochs):
        running_loss = 0.0
        opt_SGD.zero_grad()
        pre = net(data.float())
        loss = loss_fun(pre,label.long().squeeze())
        loss.backward()
        opt_SGD.step()
        running_loss += loss.item()
        print("Epoch%03d: Training_loss = %.5f" % (epoch + 1, running_loss))
```

(3) 怎么看loss和acc的变化

train loss 不断下降，test loss不断下降，说明网络仍在学习;
train loss 不断下降，test loss趋于不变，说明网络过拟合;
train loss 趋于不变，test loss不断下降，说明数据集100%有问题;
train loss 趋于不变，test loss趋于不变，说明学习遇到瓶颈，需要减小学习率或批量数目;
train loss 不断上升，test loss不断上升，说明网络结构设计不当，训练超参数设置不当，数据集经过清洗等问题。

(4) MaxPooling

pytorch中池化层MaxPool2d函数

(5) 卷积、BN、激活函数的顺序

卷积->BN->激活函数

(6) BN

torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

num_features：通道数

eps：防止除 0

(7) dropout

out = nn.functional.dropout(out, p=0.5, training=self.training)

(8) BN在训练和测试时的差别

对于BN，在训练时，是对每一批的训练数据进行归一化。使用BN的目的就是每个批次分布稳定。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。

而在测试时，比如进行一个样本的预测，就并没有batch的概念，因此，这个时候用的均值和方差是全量训练数据的均值和方差，也就是使用全局统计量来代替批次统计量，这个可以通过移动平均法求得。具体做法是，训练时每个批次都会得到一组（均值、方差），然后对这些数据求数学期望！每轮batch后都会计算，也称为移动平均。

(9) Dropout 在训练和测试时都需要嘛?

Dropout 在训练时采用，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。

而在测试时，应该用整个训练好的模型，因此不需要dropout。