深度学习学习笔记

本笔记为Andrew和Udacity的深度学习课程所做，鉴于Andrew课程已有完整笔记，这里只做对笔者而言重要的笔记，可能对别的读者没有太大用处。 <https://github.com/Parker-Lyu/DeepL_Andrew_And_Uda>项目说明上有相关作业及笔记的链接。

本笔记的顺序按照Andrew的课程来做，Uda的课程作为项目补充。

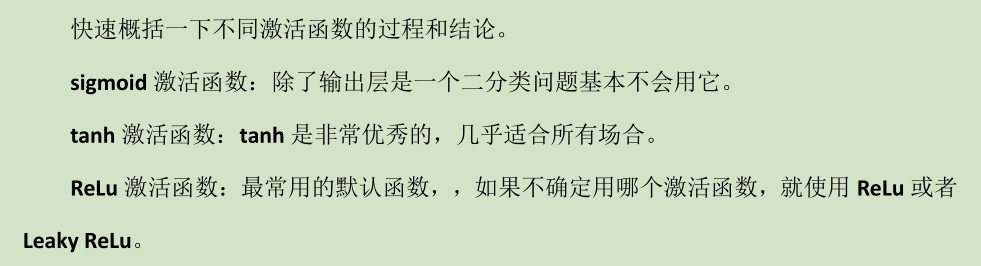
# 第一课 神经网络和深度学习

这部分主要讲了网络基础，关于梯度下降，自己写个网络，写出前向传播的公式，并且按照链式法则写出反向传播的公式，就可以用代码实现了。关于需要转置的问题，自己记录好矩阵纬度就ok了。

反向传播的代码实现见notebook。

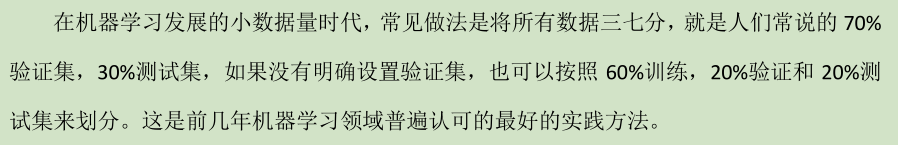
## 激活函数

为什么需要非线性激活函数？

如果没有非线性激活函数，深层神经网络可以看成线性函数的组合，计算一下整个网络就是一个单层的线性函数。

# 第二课 改善深层神经网络

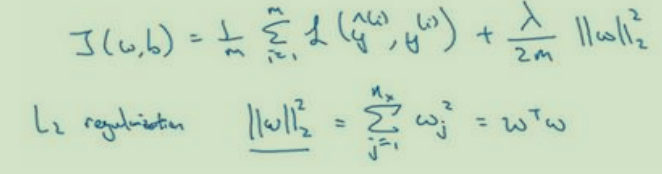
## 训练验证测试集



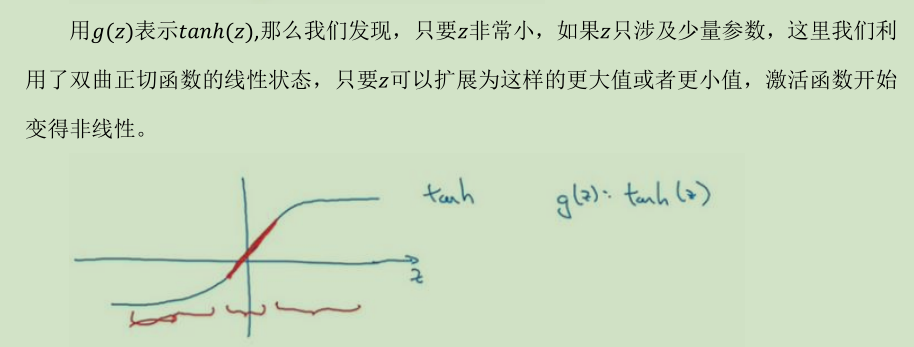
现在我们的数据量都很大，如果我有100W条数据，那么98w用来训练，1W作为验证，1W作为测试；

## 正则化

### L2范数/权重衰减



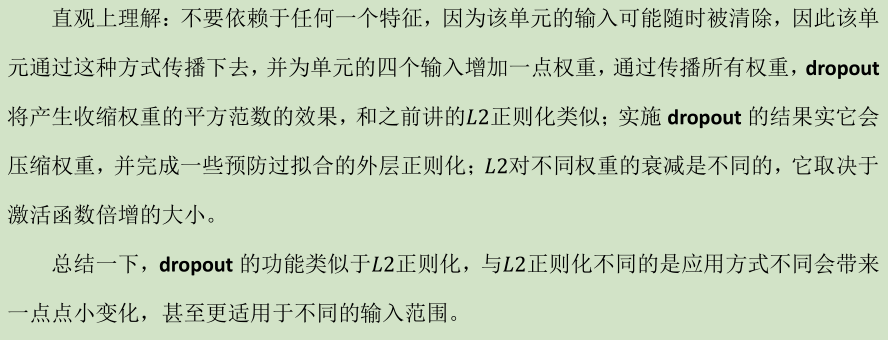
解释L2正则化，如果正则化参数lambda设置的足够大，所有w都接近于0，z也会接近于0；



这样，网络的激活函数会接近线性函数；之前讲过，线性激活函数的网络不过是个线性函数，这样，网络逐渐向线性回归靠拢，渐渐丧失过拟合的能力；

### dropout

drop用来解决过拟合，如果没有过拟合的现象，不要用。



**数据集要服从统一分布**

### 其它解决过拟合的正则化方法

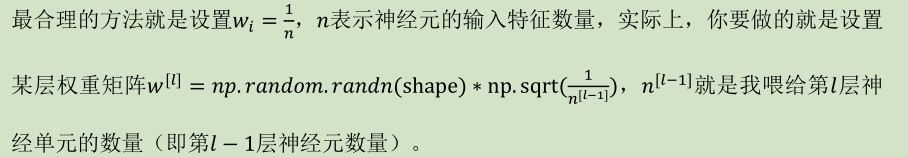
数据扩增：对称、翻转、裁切、加噪音

## 归一化输入特征有利于网络训练

## 神经网络权重初始化：

防止梯度消失/爆炸；

神经网络的计算实际上是这样的，为了防止z值过大或者过小，n越大，我们通常希望w越小，因为z是wx的和；



这个公示的意思实际上是，把标准正太分布除以的平方根，结果生成均值位0，方差为的矩阵；

其实，总结来讲，如果使用relu激活函数，设置方差为，

tanh作为激活函数，作为方差；

## 优化算法

mini-batch

优化方法：Momentum/RMSprop/Adam/Learning rate decay;

## 超参数调试：

两种调测参数的策略：

数据量超大而没有足够的计算资源时，在它实验时逐渐改良，不断调整。

有足够的计算资源时，快速实验；

## batch归一化

做数据预处理时，已经归一化了输入数据的特征，使训练更有效率，那训练过程中，归一化节点的值也会使训练更有效率。默认推荐归一化z值，即先归一化再激活。

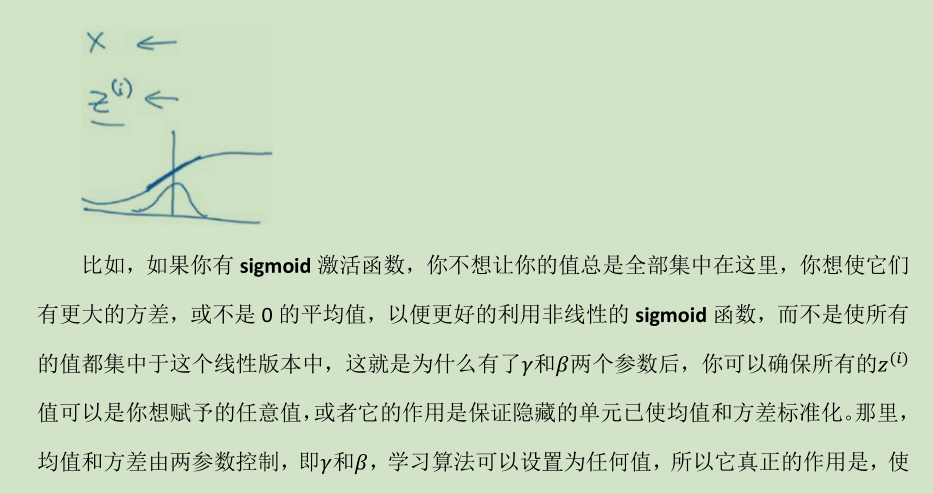
归一化不是把z固定为均值为0，方差为1，而是加入这个方程



其中

方程中的gama和bate是网络会自动更新学习的参数；

该方程的意义：



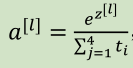
Andrew的课中讲到，因为batch-normalization操作，会重置z的均值，所以z = wx + b这一步计算中，b对z值的任何改变都会被normalization操作归一化，所以b是可以取消的。具体解释一下，batch-normalization这个思想来源于初始数据的归一化可以加快训练，也就是说，对于网络中的每个z，（每一层网络可能有n个z节点），只计算该批次数据中该z的std，mean，并进行归一化，而不是把该批次的该层所有z节点数据进行归一化。

batch-normalization有轻微的正则化效果，因为这个算法使基于mini-batch的，而每一个mini-batch的数据分布不会完全一样，这就使得每一批次的数据std，mean值不一样，就像dropout的操作一样，不使得网络过分依赖某一种数据分布；

训练时会使用mini-batch，但是实际使用网络时可能就是一次一条数据了，这时一条数据的std，mean就没有什么意义了，所以，而对于每一层的每一个z节点的std, mean，都可以用移动平均（指数加权平均）的方法来计算，这样即便在测试时，也可以使用batch-normalization。

## softmax

激活函数

为了使输出之和为1，

# 第三课 结构化机器学习项目（机器学习策略）

## 单一数字评估指标

### 查准率和查全率：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 预测 | |
|  |  | 1 | 0 |
| 实际 | 1 | TP | FN |
| 0 | FP | TN |

查准率precision，指预测为正的结果有多少是真正值，

TP/(TP+FP)

查全率recall，对所有结果为正的数据正确预测出了多少

TP/(TP+FN)

综合查准率查全率的比分，F1



### AUC/ROC

受试者工作特征曲线 （receiver operating characteristic curve，简称ROC曲线），得此名的原因在于曲线上各点反映着相同的感受性，它们都是对同一信号刺激的反应，只不过是在两种不同的判定标准下所得的结果而已

AUC定义为ROC曲线下的面积。

ROC是以假阳性概率（False positive rate）为横轴，真阳性（True positive rate）为纵轴所组成的坐标图，和受试者在特定刺激条件下由于采用不同的判断标准得出的不同结果画出的曲线。

真阳性概率，实际意义为预测出的真阳性占真实数据的阳性比例；

假阳性概率，实际意义为预测出的假阳性概率占阴性数据的比例；

TPR = TP/(TP+FN) FPR = FP/(FP+TN)

## 可避免偏差

指目前训练集误差与贝叶斯最有误差之间的差距，训练到一个阶段，看一下可避免误差以及方差的提升空间，选择优化偏差还是方差。

## 误差分析

查看错误数据，分析错误原因，针对数据量最大的错误类型进行修正，以最小的劳动获得最大的性能提升；

## 错误标注数据

深度学习算法对训练集的随机误差是相当健壮的，

但是系统误差对算法的影响会很大；

是否修改错误标注的数据，要进行一下错误分析，看一下错误标注数据所占比例；

如果要修改数据，测试集也要修改，要保证开发集、测试集的数据来自同一分布；

## 快速搭建第一个系统，并进行迭代

定一个目标，快速实现一个网络，做偏差/方差分析，迭代网络。

## 训练/开发/测试数据来自不同分布

### 在不同的划分上进行训练并测试

假设要开发一个手机应用，用户上传手机拍摄的照片，算法识别照片是否有猫。手上只有1W张用户上传的照片和20W张从网页爬取的照片，那么，为了实际业务场景着想，算法应该针对实际数据分布。所以数据划分如下：

205000训练，2500验证(全是用户上传)，2500测试(全是用户上传)。

这样会有训练/开发数据集不匹配的问题，下面讲如何评估并解决这个问题

### 不匹配数据划分的偏差/方差

在不匹配数据划分上训练数据，如果验证集上数据表现很差，也就是说方差很大，可能的问题原因会有两个：1、数据不匹配造成的，2、网络泛化能力差。只有找到原因才能够提升网络性能。

这个时候就要设置训练-开发集。在训练集中随机划分出一部分数据作为训练开发集，评估网络性能，而**不参与训练**。

那么我们现在把数据划分为了4个部分,其错误率如下：

|  |  |
| --- | --- |
| Optimal Bayes error(最优贝叶斯误差) | 1% |
| Train error(训练误差) | 2% |
| Train-dev error(训练-开发(验证)误差) | 9% |
| Dev error(开发/验证误差): | 10% |
| Test error(测试误差): | 10% |

最优贝叶斯误差和训练误差之间的差距代表了偏差。

训练误差和训练-开发误差的数据来自同一分布，他们之间的差代表了网络方差大小，即网络泛化能力。

训练-开发误差和开发误差的差值代表了数据不同分布造成的误差，这个是由数据不同分布造成的，下一节讲这个问题怎么解决。

### 定位数据不匹配

做人工误差分析，弄清楚训练集和开发集的数据差异；

把训练集模拟得更像开发集，收集更多跟开发集相似的数据。

可以选择合成数据，但是注意合成数据的原材料不要过于单一。

## 迁移学习

在迁移学习的概念中，网络的训练一般有两个阶段，第一个阶段是网络从随机初始化的参数开始训练，第二个阶段是更换目标与数据集训练，这样第一个阶段称为预训练pre-training，第二个阶段称为微调fine tuning；

迁移学习的意义，提升网络效果。

迁移学习的经验，来自无人驾驶Term1

从头训练一个网络要经历设计、训练、调整这些步骤，会比较慢，而迁移学习可以很快达到一个比较好的目标。根据新数据量及与原始数据的相似程度，迁移学习面临4种情况。

1、新数据集小，与原始数据分布较像；

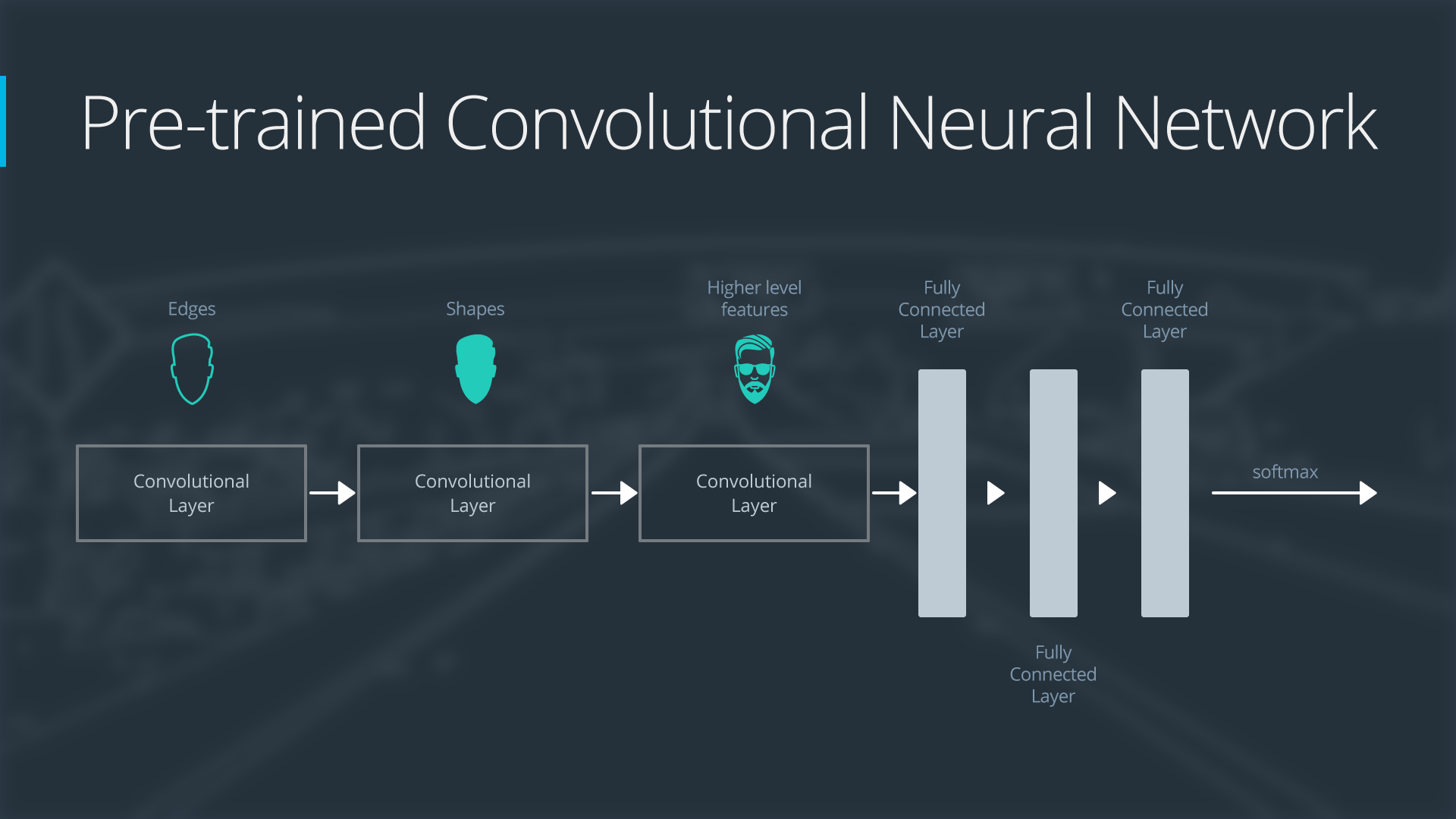
2、数据量小，差异大

3、数据量大，差异小

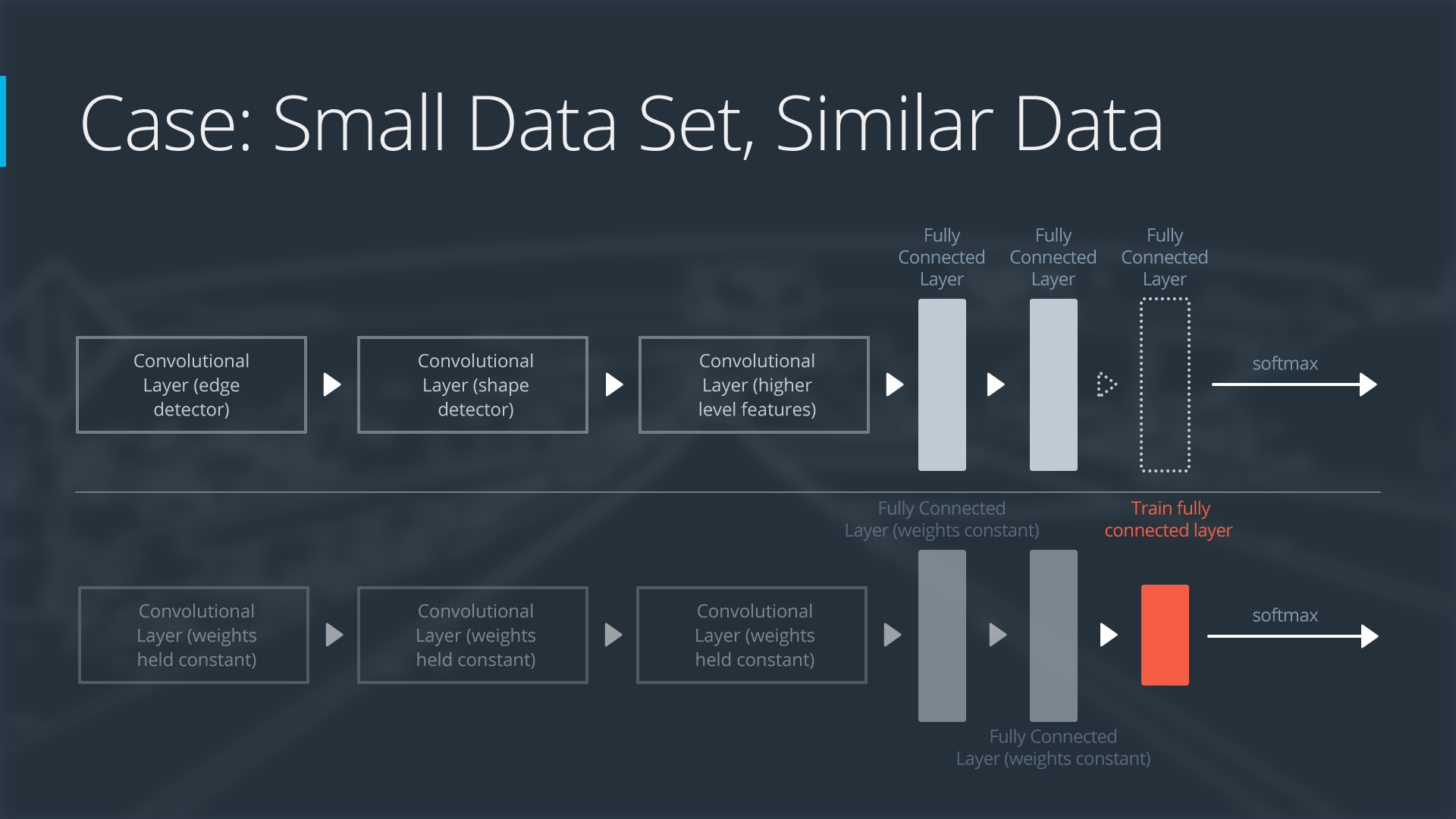
4、数据量大，差异大

下面是uda的思路：

原始网络的样子：



### 1、small-similar

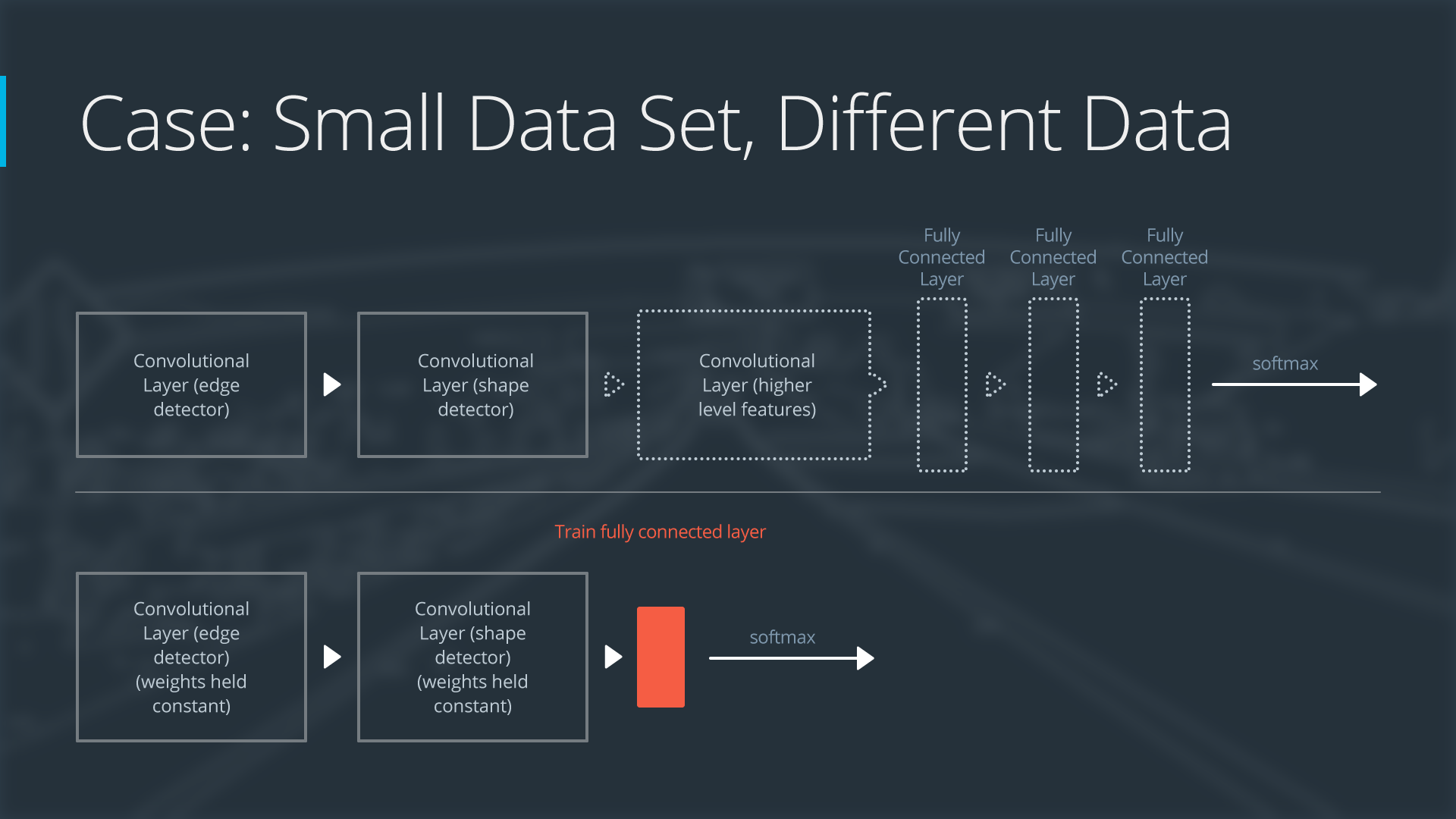


替换掉原始网络的最后一层，并冻结前面的参数，只训练最后一层，原因如下：

为避免小数据集过度拟合，原始网络的权重将保持不变，而不是重新训练权重。

由于数据集相似，每个数据集的图像将具有相似的更高级别的特征。因此，大多数或所有预先训练的神经网络图层都已经包含了有关新数据集的相关信息，因此应予以保留。

### 2、Small-Different



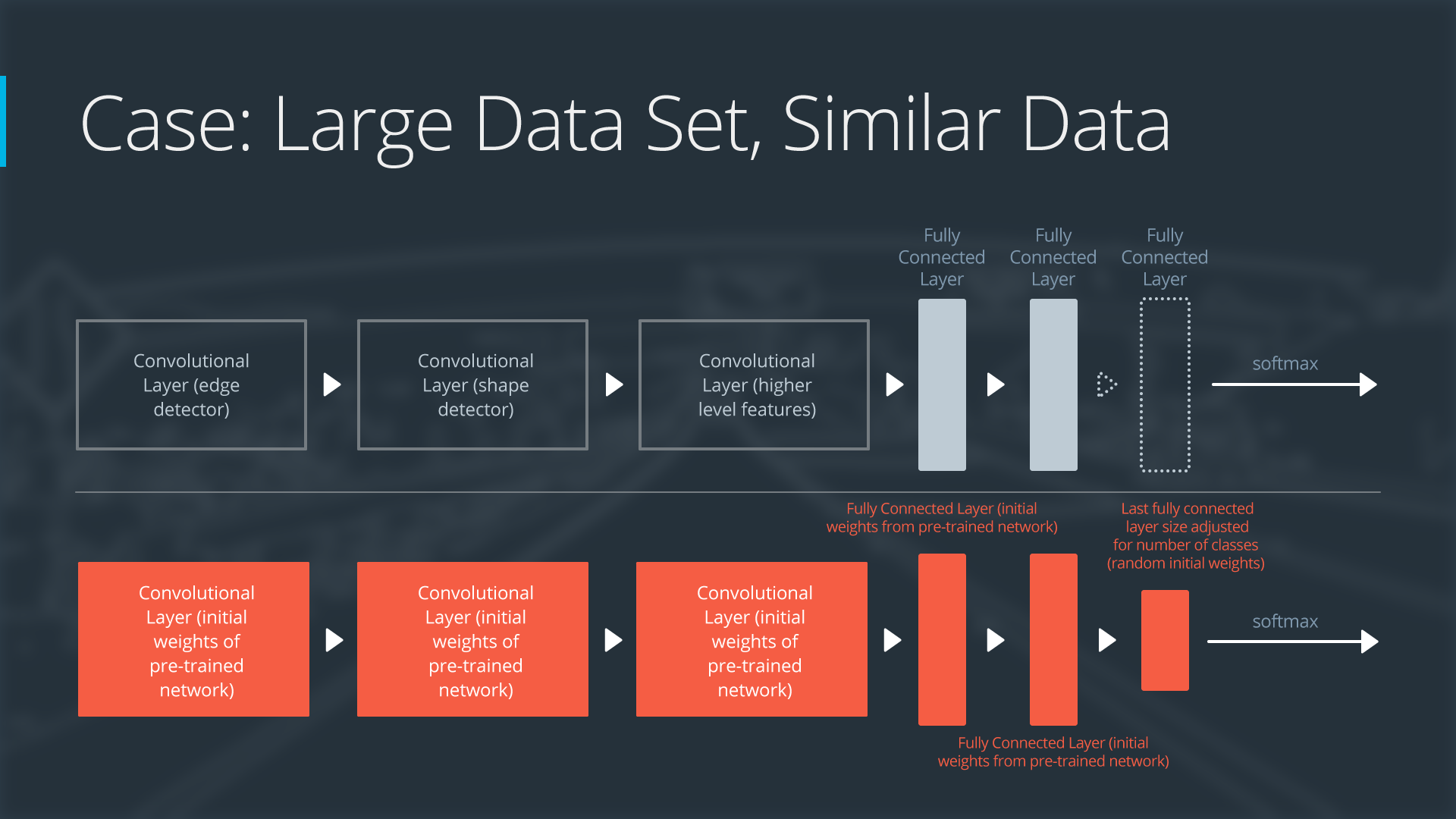
如果新数据集较小并且与原始训练数据不同：

* 切断网络开始附近的大部分预先训练的层
* 向剩余的预先训练的图层添加一个与新数据集中的类数相匹配的全新连接图层
* 随机化新的完全连接层的权重; 冻结来自预先训练的网络的所有权重
* 训练网络以更新新的完全连接层的权重

由于数据集很小，过度拟合仍然是一个问题。为了防止过拟合，原始神经网络的权重将保持不变，就像第一种情况一样。

但原始训练集和新数据集不共享更高级别的功能。在这种情况下，新网络将只使用包含较低级别功能的图层。

### 3、Large-Similar



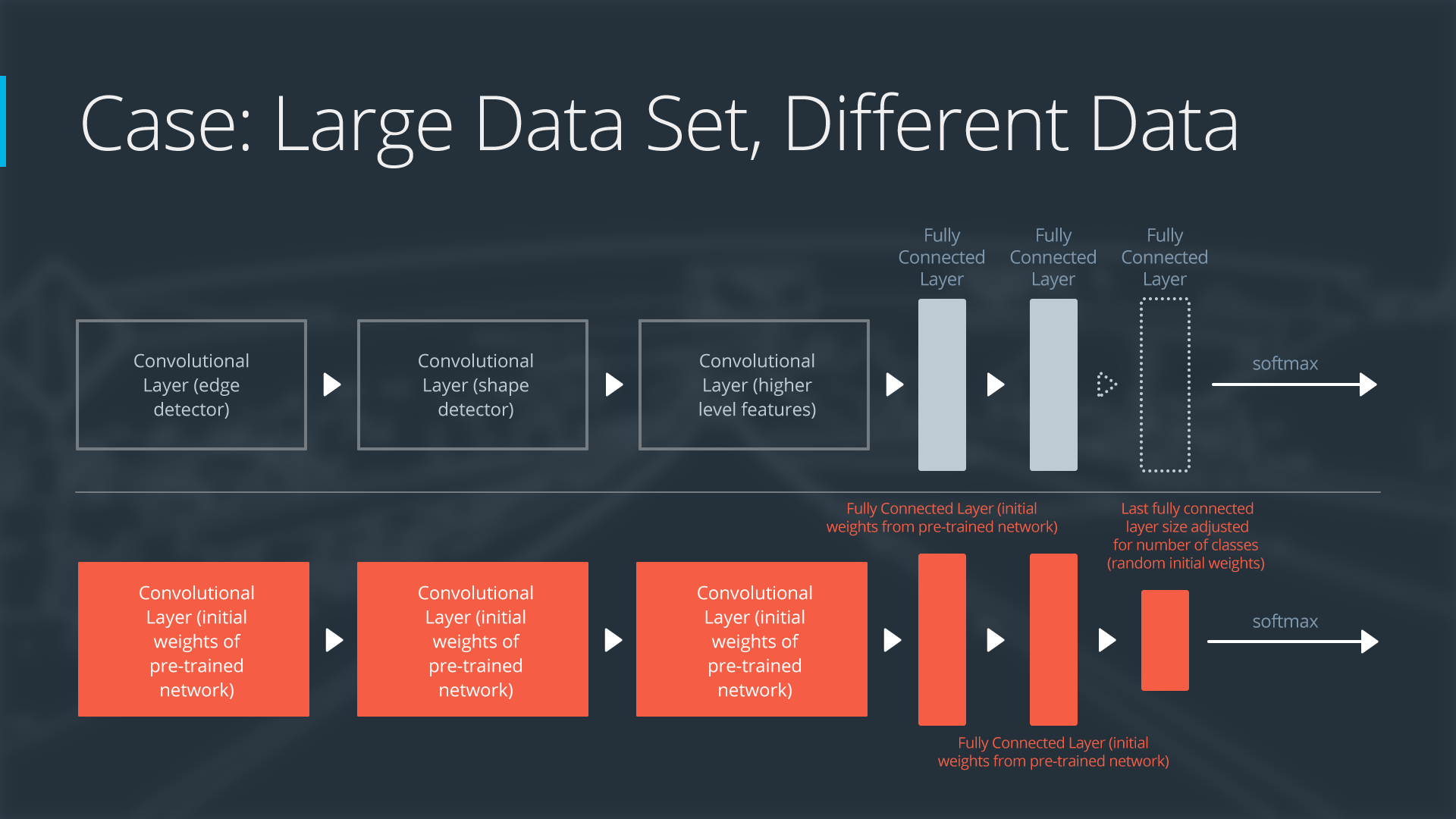
如果新数据集很大并且与原始训练数据相似：

* 删除最后一个完全连接的图层，并替换为与新数据集中的类数匹配的图层
* 随机初始化新完全连接层中的权重
* 使用预先训练的权重初始化其余权重
* 重新训练整个神经网络

在对大型数据集进行培训时，过度拟合并不是那么重要; 因此，你可以重新训练所有的权重。

由于原始训练集和新数据集共享更高级别的特征，因此也使用整个神经网络。

### 4、Large-Different



如果新数据集很大并且与原始训练数据不同：

* 删除最后一个完全连接的图层，并替换为与新数据集中的类数匹配的图层
* 从头开始用随机初始化权重重新训练网络
* 或者，您可以使用与“大而相似”的数据案例相同的策略

即使数据集与训练数据不同，从预先训练的网络初始化权重可能会使训练速度更快。所以这种情况与大量类似数据集的情况完全相同。

如果使用预先训练的网络作为起点并不能产生成功的模型，另一种选择是随机初始化卷积神经网络权重并从头开始训练网络。

### 总结

是否固定前边层的权重，取决与数据量，数据量小，为避免过拟合就要固定前面的权重。

是否删除

## 多任务学习

多任务学习可以适应图像只有部分物体有标签的情况；在做loss函数的时候，只对0和1的数据求loss的和，忽略未知项。

三点：

1、训练的一组任务，可以使用相同的低层次的特征，表现会好。

2、对每个任务的数据量相似（这个不是绝对的），做某个任务的时候，这些任务可以帮助网络的低层次特征构建，进而帮助到别的任务。

3、多任务网络，如果网络足够大，其性能是不会比多个单任务网络差的。

## 端到端的深度学习

### 优点：

一、让数据说话，而不是引入人类见解，容易造成偏见

二、需要手工设计的组件很少，简化工作流程。

### 缺点

一、需要大量数据

二、排除了可能有用的人工设计组件。