# Part1-2：the mathematical building blocks of neural networks

本章包括

1、一个神经网络的例子

2、张量和张量操作

3、神经网络如何通过反向传播和梯度下降来学习

## 一、[一个神经网络的例子](https://github.com/592595/DeepLearning/blob/master/2.1-a-first-look-at-a-neural-network.py)

使用Python库Keras学习对手写数字进行分类的神经网络（将手写数字（28\*28px）的灰度图像分为10个类别：0-9；使用MNIST数据集，含有6000张测试图像，10000张训练图像）

1. 在Keras中加载MNIST数据集

from keras.datasets import mnist

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

其中图像编码为numpy数组，label是一个标签数组（范围0-9），图像标签一一对应，流程：1）向神经网络提供训练数据；2）network学习相关的图像和标签；3）让network为test\_image生成预测、验证预测是否与test\_label中的标签匹配，再次建立网络

1. 网络架构

from keras import models

from keras import layers

network = models.Sequential()

network.add(layers.Dense(512, activation='relu', input\_shape=(28 \* 28,)))

network.add(layers.Dense(10, activation='softmax'))

深度学习模型就像是数据处理的筛子由一系列日益完善的数据过滤器—layers组成

选择三件事，作为编译步骤的一部分：

1. loss function

衡量模型在训练上的表现

1. An optimizer 优化器

自行更新的机制

1. 要监控的指标

在这里只关心准确性（正确分类的图像的分数）

1. 编译步骤

network.compile(optimizer='rmsprop',

loss='categorical\_crossentropy',

metrics=['accuracy'])

训练前，将数据重新整形为网络所需要的形状并对其进行缩放，时期在[0,1]内来预处理数据

uint8 255 (6000,28,28)矩阵🡪float [0,1] (6000,28\*28)数组

1. 准备数据图像

train\_images = train\_images.reshape((60000, 28 \* 28))

train\_images = train\_images.astype('float32') / 255

test\_images = test\_images.reshape((10000, 28 \* 28))

test\_images = test\_images.astype('float32') / 255

1. 准备标签

from keras.utils import to\_categorical

train\_labels = to\_categorical(train\_labels)

test\_labels = to\_categorical(test\_labels)

都准备好后，在Keras中通过调用网络的拟合方法来完成模型与训练数据相应匹配：

network.fit(train\_images, train\_labels, epochs=5, batch\_size=128)

显示训练数据在网络丢失和网络对训练数据的准确性、输出在测试数据上的准确性：

test\_loss, test\_acc = network.evaluate(test\_images, test\_labels)

print('test\_acc:', test\_acc)

## 二、神经网络的数据表示—张量

张量是数据的容器 - 几乎总是数字数据。

1. 标量（0D张量）

仅包含一个数字的张量称为标量（或标量张量，或0维张量，或0D张量）。 在Numpy中，float32或float64数字是标量张量（或标量数组）。标量张量有0轴（ndim == 0）。

1. 向量（1D张量）

数组数组称为向量或1D张量，1D张量恰好具有一轴。

注意区分轴与维度，例如：x = np.array([12, 3, 6, 14])； 5D矢量仅具有一个轴并且沿其轴具有五个维度，而5D张量具有五个轴（并且沿着每个轴可以具有任意数量的维度）。

1. 矩阵（2D张量）

向量数组是矩阵或2D张量，矩阵有两个轴（通常称为行和列）。

1. 3D张量和高维张量

x = np.array([[[5, 78, 2, 34, 0],

[6, 79, 3, 35, 1],

[7, 80, 4, 36, 2]],

[[5, 78, 2, 34, 0],

[6, 79, 3, 35, 1],

[7, 80, 4, 36, 2]],

[[5, 78, 2, 34, 0],

[6, 79, 3, 35, 1],

[7, 80, 4, 36, 2]]])

在视觉上可以将其解释为数字的立方体。通过在阵列中打包3D张量，可以创建4D张量，依此类推。

1. 三个关键属性：
2. 轴数（等级）：ndim
3. 形状：shape
4. 数据类型：dtype
5. 数据批量的概念

通常，在深度学习中遇到的所有数据张量中的第一个轴（轴0，因为索引从0开始）将是样本轴（有时称为样本维度）。 在MNIST示例中，样本是数字的图像。 此外，深度学习模型不会立即处理整个数据集; 相反，他们将数据分成小批量。 当考虑这种批量张量时，第一轴（轴0）称为批量轴或批量维度。

1. 数据张量的真实示例，常见的如下：
2. 矢量数据 - 形状的2D张量（样本，特征）

一个人的精算数据集，我们考虑每个人的年龄，邮政编码，和收入。 每个人可以被表征为3个值的向量，因此可以将100,000个人的整个数据集存储在2D张量形状（100000,3）中。

1. 时间序列数据或序列数据 - 形状的3D张量（样本，时间步长，特征）

按照惯例，时间轴始终是第二轴（索引1的轴）。

股票价格数据集。 每分钟，我们存储当前的股票价格，过去一分钟的最高价格和过去一分钟的最低价格。因此，每分钟编码为一个3D矢量，整个交易日编码为2D张量 形状（390,3）（在交易日有390分钟），250天的数据可以存储在3D张量的形状（250,390,3）中。 在这里，每个样本都是一天的数据

1. 图像-4D形状张量（样本，高度，宽度，通道）或（样本，通道，高度，宽度）

图像通常具有三个维度：高度，宽度和颜色深度。

1. 视频-5D形状张量（样本，帧，高度，宽度，通道）或（样本，帧，通道，高度，宽度）

## 三、张量运算

在我们的初始示例中，我们通过将Dense图层堆叠在一起来构建我们的网络。

1. 元素操作，可以在numpy中十分简单的就做好
2. 广播

广播包括两个步骤：1）轴（称为广播轴）被添加到较小的张量以匹配较大张量的ndim；2）较小的张量与这些新轴重复，以匹配较大张量的整个形状

1. np.dot(x,y)—矩阵相乘; \*则表示逐个元素相乘
2. shape()显示当前矩阵形状、reshape()改变矩阵形状

## 四、神经网络的引擎：基于梯度的优化

training loop步骤：

1）绘制一批训练样本x和相应的目标y。

2）在x上运行网络（称为前向传递的步骤）以获得预测y\_pred。

3）计算批次上网络的loss，衡量y\_pred和y之间的不匹配。

4）以略微减少此批次损失的方式**更新网络的所有权重。**

更新网络权重是最难的一个步骤，给定网络中的单个权重系数，如何计算系数是应该增加还是减少，以及增加多少？一种好的方法是利用网络中使用的所有操作都是可微分的这一事实，并根据网络系数计算损耗的梯度。 然后，可以从梯度向相反方向移动系数，从而减少损失。

1、张量运算的导数：梯度

以张量作为输入的函数。 考虑输入向量x，矩阵W，目标y和损失函数损失。 可以使用W计算目标候选y\_pred，并计算目标候选y\_pred和目标y之间的loss或者 mismatch

y\_pred = dot(W, x)

loss\_value = loss(y\_pred, y)

2、随机梯度下降

给定一个可微函数，理论上可以分析地找到它的最小值：已知函数的最小值是导数为0的点，所以我们要做的就是找到所有点，这里导数变为0，并检查这些点中的哪一个函数具有最低值。应用于神经网络，这意味着在分析上找到产生最小可能损失函数的权重值的组合。 这可以通过求解W的方程梯度（f）（W）= 0来完成。从梯度更改相反方向的权重，每次丢失的次数会少一些，更新网络的所有权重的有效方法如下：

1）绘制一批训练样本x和相应的目标y。

2）在x上运行网络以获取预测y\_pred。

3）计算批次上网络的丢失，衡量y\_pred和y之间的不匹配。

4）根据网络参数计算loss的梯度（反向传递）。

5）在与梯度相反的方向上稍微移动参数 - 例如W - = step \* gradient - 从而减少批次上的loss。

为步长因子选择合理的值非常重要。如果它太小，曲线下降将需要多次迭代，并且它可能会陷入局部最小值。如果步长太大，更新最后可能会带到曲线上完全随机的位置。

注意，小批量SGD算法的一种变体是在每次迭代时绘制单个样本和目标，而不是绘制一批数据。另外，存在多种SGD变体，其在计算下一个权重更新时考虑先前的权重更新而不是仅仅查看梯度的当前值而不同。

3、反向传播算法

将链式规则应用于神经网络的梯度值的计算产生称为反向传播的算法（有时也称为反向模式区分）

## 五、对第一个例子的总结

这是输入数据

Numpy，在这里格式化为float32类型：

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

train\_images = train\_images.reshape((60000, 28 \* 28))

train\_images = train\_images.astype('float32') / 255

test\_images = test\_images.reshape((10000, 28 \* 28))

test\_images = test\_images.astype('float32') / 255

network如下：

network = models.Sequential()

network.add(layers.Dense(512, activation='relu', input\_shape=(28 \* 28,)))

network.add(layers.Dense(10, activation='softmax'))

网络编译步骤：

categorical\_crossentropy是用作学习权重张量的反馈信号的损失函数，训练阶段将尝试最小化。 这种损失的减少是通过小批量随机梯度下降发生的。 控制梯度下降的特定使用的确切规则由作为第一个参数传递的rmsprop优化器定义。

network.compile(optimizer='rmsprop',

loss='categorical\_crossentropy',

metrics=['accuracy'])

最后，训练循环：

网络将开始以128个样本的小批量重复训练数据，5次以上（对所有训练数据的每次迭代称为纪元）。 在每次迭代时，网络将根据批次上的损失计算权重的梯度，并相应地更新权重。 在这5个时期之后，网络将执行2,345个梯度更新（每个时期469个），并且网络的丢失将足够低，使得网络能够以高精度对手写数字进行分类。

network.fit(train\_images, train\_labels, epochs=5, batch\_size=128)

# Part1-3：Getting started with neural networks

本章包括：

1）神经网络的核心组件

2）对Keras的介绍

3）设置深度学习工作站

4）使用神经网络解决基本分类和回归问题

## 一、神经网络的剖析

训练神经网络围绕以下对象：

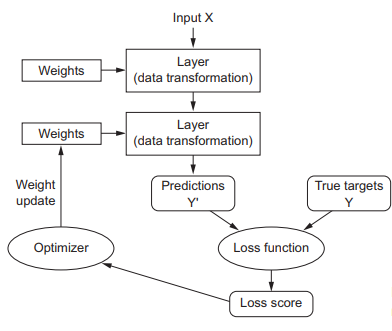
 1）图层，组合成网络（或模型）。层是数据处理模块，其将一个或多个张量作为输入并且输出一个或多个张量。存储在形状（样本，特征）的2D张量中的简单矢量数据通常由密集连接的层处理，也称为完全连接或密集层（Keras中的密集类）。 存储在形状（样本，时间步长，特征）的3D张量中的序列数据通常由诸如LSTM层的循环层处理。 存储在4D张量中的图像数据通常由2D卷积层（Conv2D）处理。

 2）输入数据和相应的目标

常见的网络拓扑结构：Two-branch networks、 Multihead networks、 Inception blocks。通过选择网络拓扑，可以将可能性空间（假设空间）约束到一系列特定的张量，将输入数据映射到输出数据。 接下来将要搜索的是这些张量操作中涉及的权重张量的一组很好的值。

 3）损失函数，定义用于学习的反馈信号。使其最小化。

 4）优化器，决定学习的进度



## 二、 Keras

目前，三个现有的后端实现是TensorFlow后端，Theano后端和Microsoft Cognitive Toolkit（CNTK）后端。使用Keras编写的任何代码都可以使用任何这些后端运行，而无需更改代码中的任何内容。

定义模型有两种方法：使用Sequential类（仅用于线性堆栈的层，这是迄今为止最常见的网络架构）或功能API（用于层的有向非循环图，它允许您构建完全任意的体系结构）

## 三、建立深度学习工作站---NVIDIA GPU。

## 四、[电影评论分类：二分分类示例](https://github.com/592595/DeepLearning/blob/master/3.4-classifying-movie-reviews.py)

学习根据评论的文本内容将电影评论分类为正面或负面。

使用IMDB数据集：来自Internet电影数据库的一组50,000个高度极化的评论。 他们分为25,000条培训评论和25,000条测试评论，每组评估50％为负，50％为正面评价。

### 1、IMDB数据集

1）Loading the IMDB dataset

### 2、准备数据

1）Encoding the integer sequences into a binary matrix

因为无法将整数列表提供给神经网络，将列表转换为张量的两种方法：1） 填充列表，使它们都具有相同的长度，将它们转换为整数张形状（samples，word\_indices），然后将网络中的第一层用作处理此类整数张量的层（嵌入层）；2）对您的列表进行One-hot encode，将其转换为0和1的向量。这里使用的是这个。将整数序列编码为二进制矩阵。

### 3、构建网络

输入数据是向量，标签是标量（1和0）：这是将遇到的最简单的设置。Dense(16,activation='relu')，传递给每个Dense层（16）的参数是该层的隐藏单元的数量。 隐藏单元是图层的表示空间中的维度。

关于这样一堆Dense图层，有两个关键的架构决策：1）要使用多少层；2）每层选择多少个隐藏单位；这里使用了两个中间层，每个层有16个隐藏单元，第三层将输出关于当前评论情绪的标量预测。中间层将使用relu作为其激活功能，最后一层将使用sigmoid激活以输出可执行性（得分在0和1之间，越靠近1越积极）

1. The model definition
2. Compiling the model
3. Configuring the optimizer
4. Using custom losses and metrics

### 4、验证方法

1）Setting aside a validation set

1. Training your model

3）Plotting the training and validation loss

1. Plotting the training and validation accuracy
2. Retraining a model from scratch

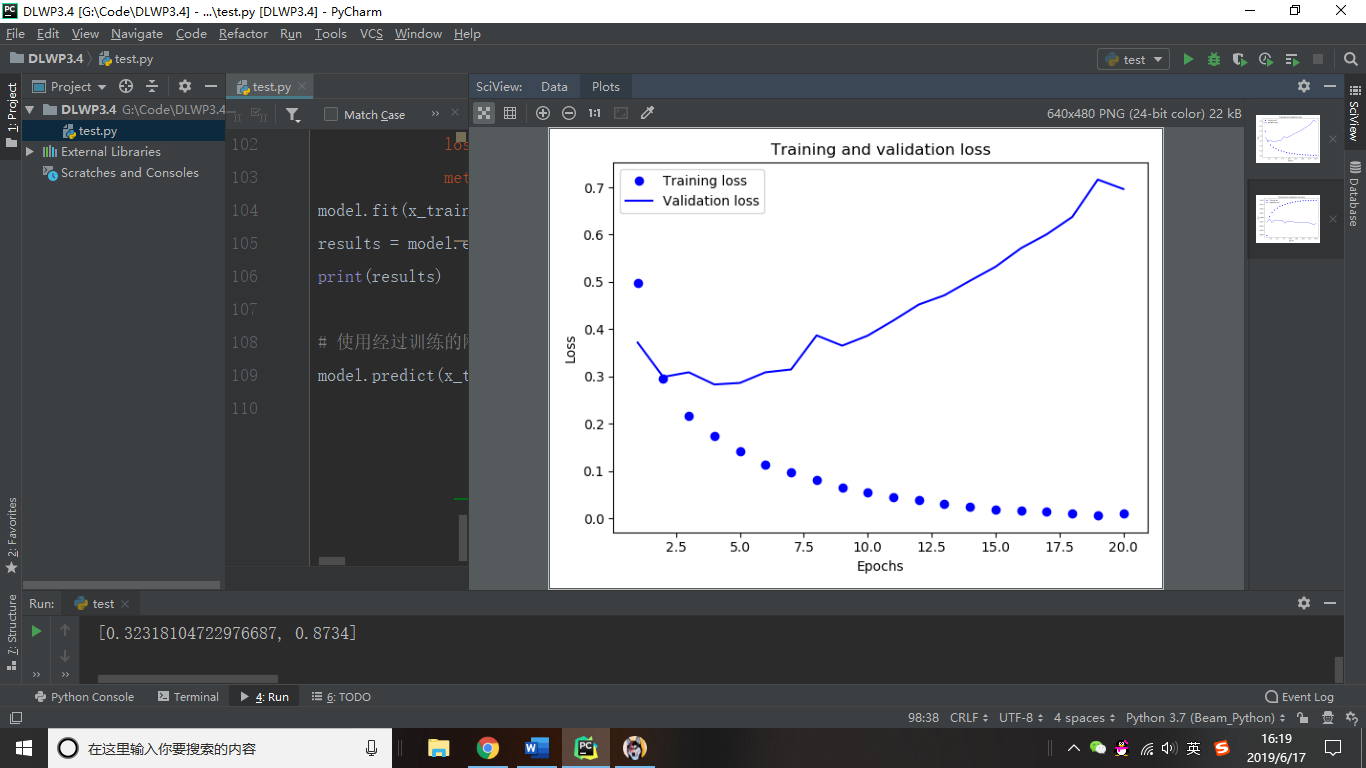
### 5、使用经过训练的网络生成对新数据的预测

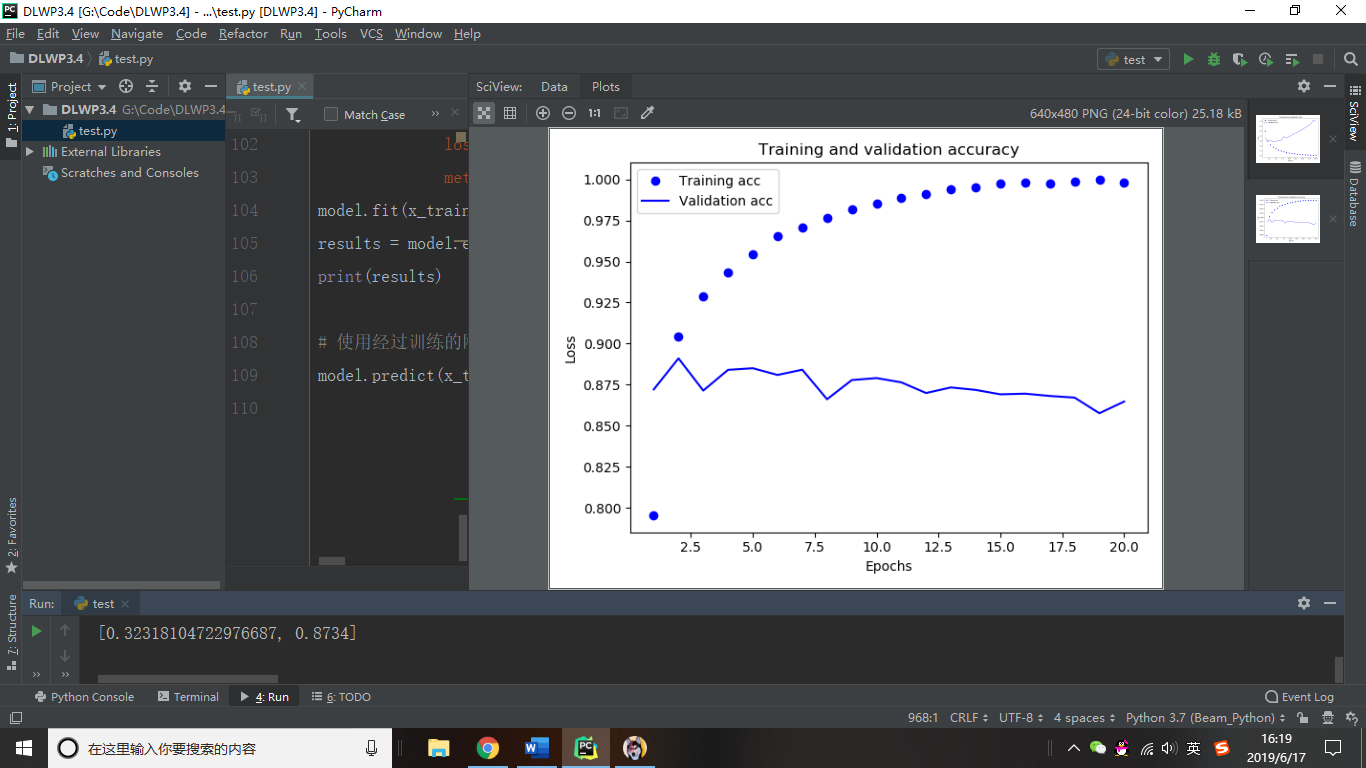
### 6、进一步实验

### 7、总结

1. 通常需要对原始数据进行相当多的预处理，以便能够将其作为张量传送到神经网络中。单词序列可以编码为二进制向量，但也有其他编码选项。
2. 具有relu激活的密集层堆栈可以解决各种问题（包括情感分类），可能经常使用。
3. 在二分分类问题（两个输出类）中，网络应该以一个单元和一个sigmoid激活的Dense层结束：网络的输出应该是0到1之间的标量，编码概率。
4. 在二分分类问题上使用这样的标量sigmoid输出，应该使用的损失函数是binary\_crossentropy。
5. 无论遇到什么问题，rmsprop优化器通常都是一个不错的选择。
6. 随着模型的训练数据越来越好，神经网络最终开始过度拟合，并最终获得越来越糟糕的数据，这些数据是模型以前从未见过的。 务必始终监控训练集之外的数据的性能。

运行结果：





## 五、[新闻专线分类：多类分类示例](https://github.com/592595/DeepLearning/blob/master/3.5-classifying-newswires.py)

这是一个single-label, multiclass classification的例子

### Reuters数据集

Reuters数据集是一个简单，广泛使用的文本分类数据集。 有46个不同的主题; 某些主题比其他主题更具代表性，但每个主题在训练集中至少有10个示例。

1. Loading the Reuters dataset
2. Decoding newswires back to text

### 准备数据

1. Encoding the data

要对标签进行矢量化，有两种方法：可以将标签列表转换为整数张量，也可以使用one-hot encoding（也叫做分类编码，经常使用）。除此外，在Keras中有一种内置的方法。（from keras.utils.np\_utils import to\_categorical）

### 构建网络

1. Model definition
2. Compiling the model

### 验证方法

1. Setting aside a validation set
2. Training the model
3. Plotting the training and validation loss
4. Plotting the training and validation accuracy
5. Retraining a model from scratch

### 生成对新数据的预测

1. Generating predictions for new data

### 处理标签和损失的不同方式

处理标签和损失的不同方式，我们之前提到过，编码标签的另一种方法是将它们转换为整数张量，这种方法唯一会改变的是损失函数的选择，从头开始重新训练模型中使用的损失函数是ategorical\_crossentropy,期望标签遵循分类编码。但是对于整数标签，应该使用sparse\_categorical\_crossentropy。

### 具有足够大的中间层的重要性

1. 一个具有信息瓶颈的模型的例子，以4维为例

### 进一步实验

1）尝试使用更大或更小的层：32个单位，128个单位，依此类推。

2）尝试使用单个隐藏层或三个隐藏层

### 9、总结

1）如果尝试在N个类别中对数据点进行分类，则网络应以大小为N的Dense层结束。

2）在单标签，多类别分类问题中，网络应以softmax激活结束，以便在N个输出类别上输出概率分布。

 3）Categorical crossentropy几乎总是我们应该用于此类问题的损失函数。它最小化了网络输出的概率分布与目标的真实分布之间的距离。

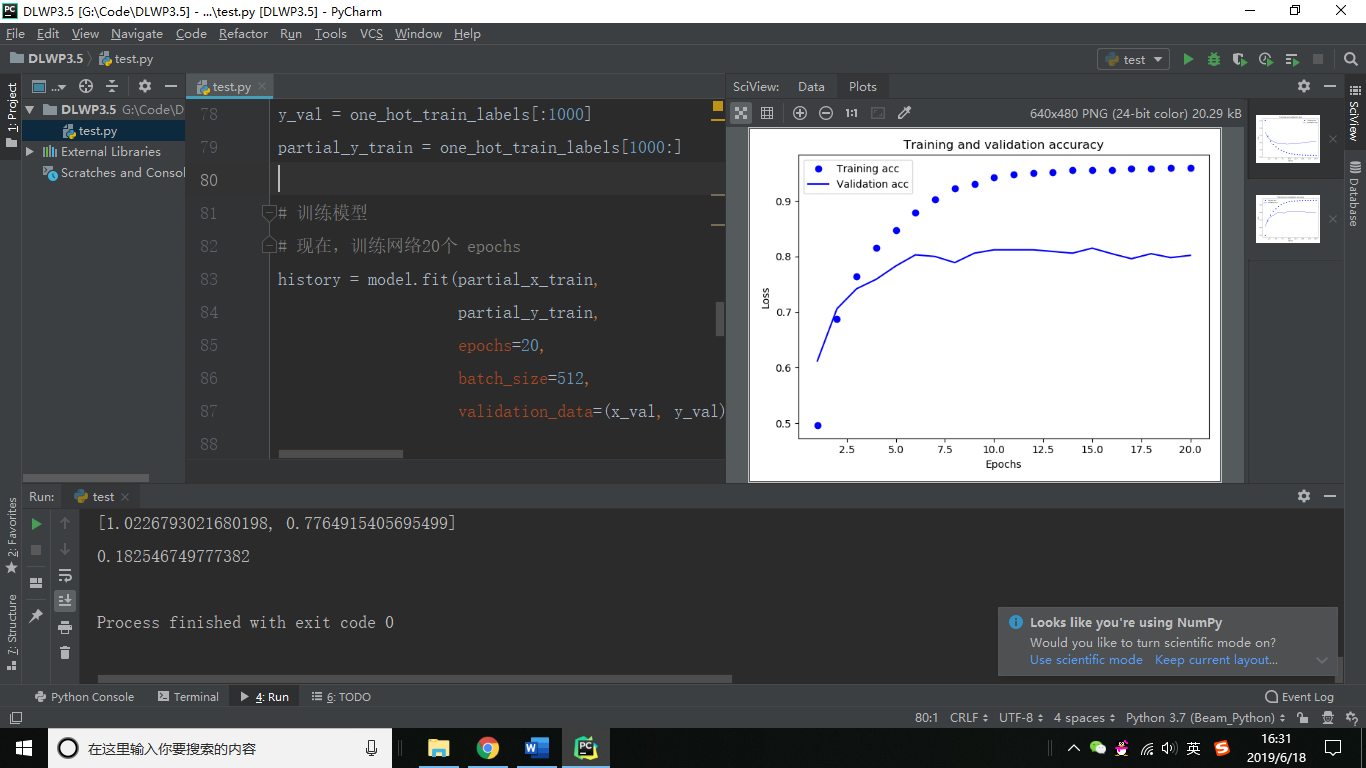
 4）在多类分类中有两种处理标签的方法：a) 通过分类编码（也称为单热编码）对标签进行编码，并使用categorical\_crossentropy作为丢失函数; b) 将标签编码为整数并使用arse\_categorical\_crossentropy损失函数

5）如果需要将数据分类为大量类别，则应避免由于中间层太小而在网络中造成信息瓶颈。

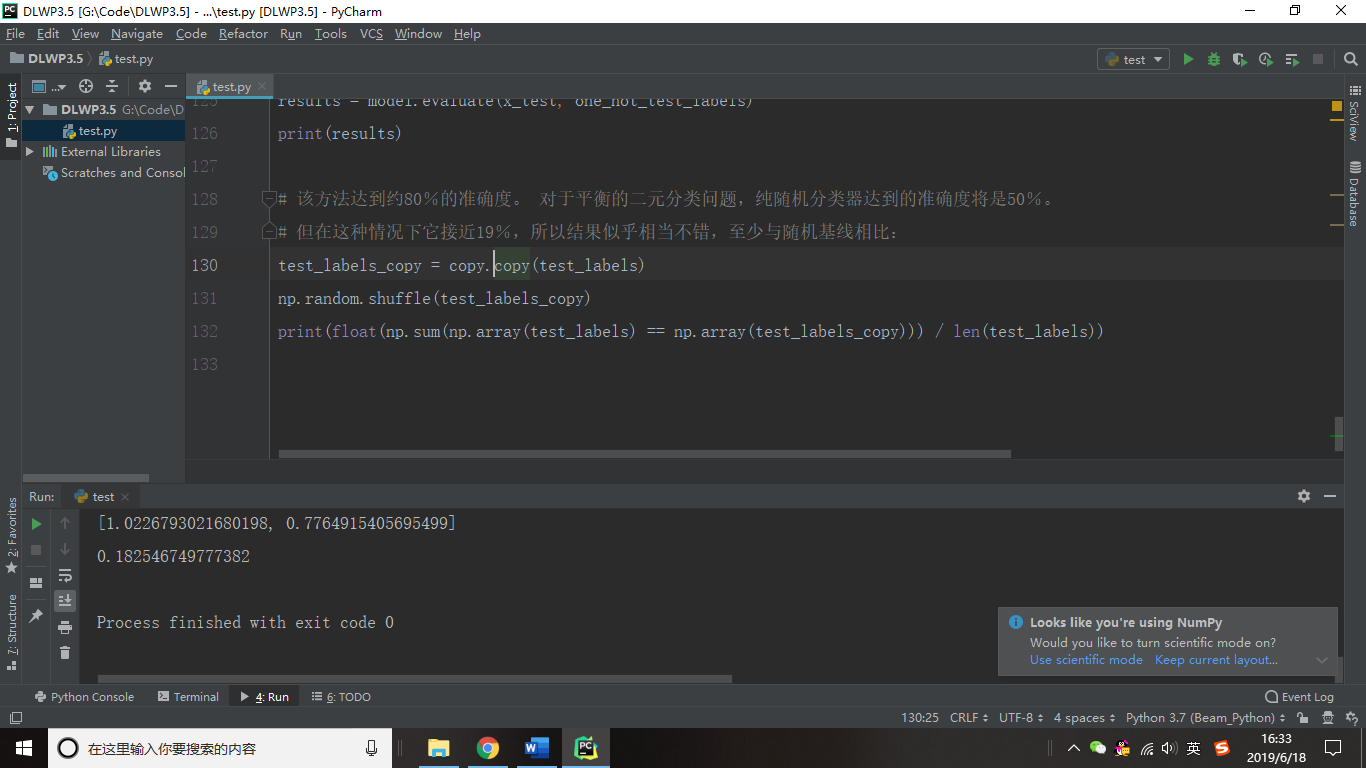
运行结果：

显示loss 和 accuracy 曲线：





显示准确度和与random baseline的对比：



## [六、预测房价：一个回归的例子](https://github.com/592595/DeepLearning/blob/master/3.6-predicting-house-prices.py)

前两个示例被认为是分类问题，其目标是预测输入数据点的单个离散标签。 另一种常见类型的机器学习问题是回归，它包括预测连续值而不是离散标签。

注意:不要混淆回归和算法逻辑回归。 逻辑回归不是回归算法 - 它是一种分类算法。

### 波士顿住房价格数据集--Loading the Boston housing dataset

本次采用的数据集具有相对较少的数据：仅506，在404个训练样本和102个测试样本之间分配。 输入数据中的每个功能（例如，犯罪率）都有不同的比例。目标是自住房屋的中位数，价值数千美元

### 准备数据---Normalizing the data

将神经网络输入所有采用不同范围的值都是有问题的，处理此类数据的一种广泛的最佳实践是进行特征标准化：对于输入数据中的每个特征（输入数据矩阵中的一列），减去feature的平均值并除以标准偏差，以便该要素以0为中心并具有单位标准差。 这很容易在Numpy中完成。

注意，使用训练数据计算用于归一化测试数据的量。 不应该在工作流程中使用在测试数据上计算的任何数量，即使对于像数据规范化这样简单的事情也是如此。

### 构建网络---Model definition

拥有的训练数据越少，过度拟合就越差，使用小型网络是缓解过度拟合的一种方法。

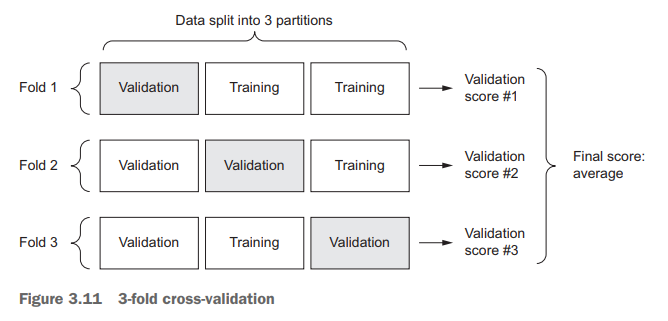
网络以单个单元结束而不激活（它将是线性层）。 这是标量回归的典型设置（尝试预测单个连续值的回归）。应用激活函数会限制输出可以采用的范围， 例如，如果将sigmoid激活函数应用于最后一层，网络只能学习预测0到1之间的值。这里，因为最后一层是纯线性的，所以网络可以自由地学习预测任何范围内的值。

对于回归问题，广泛使用的损失函数：mse loss函数 - 均方误差编译网络，即预测和目标之间差异的平方。

 在训练期间监控新指标：平均绝对误差（MAE）。 它是预测和目标之间差异的绝对值。 例如，此问题的MAE为0.5意味着您的预测平均减少500美元。

### 使用K-fold验证来验证我们的方法

1. K-fold validation



它包括将可用数据拆分为K个分区（通常为K = 4或5），实例化K个相同模型，并在评估剩余分区时对K-1分区进行训练。所用模型的验证分数是获得的K验证分数的平均值。

1. Saving the validation logs at each fold
2. Building the history of successive mean K-fold validation scores
3. Plotting validation scores
4. Plotting validation scores, excluding the first 10 data points
5. Training the final model

### 5、总结

1）使用与我们用于分类的不同的损失函数进行回归。 均方误差（MSE）是通常用于回归的损失函数。

2）用于回归的评估指标与用于分类的评估指标不同; 自然地，准确性的概念不适用于回归。 常见的回归度量是平均绝对误差（MAE）。

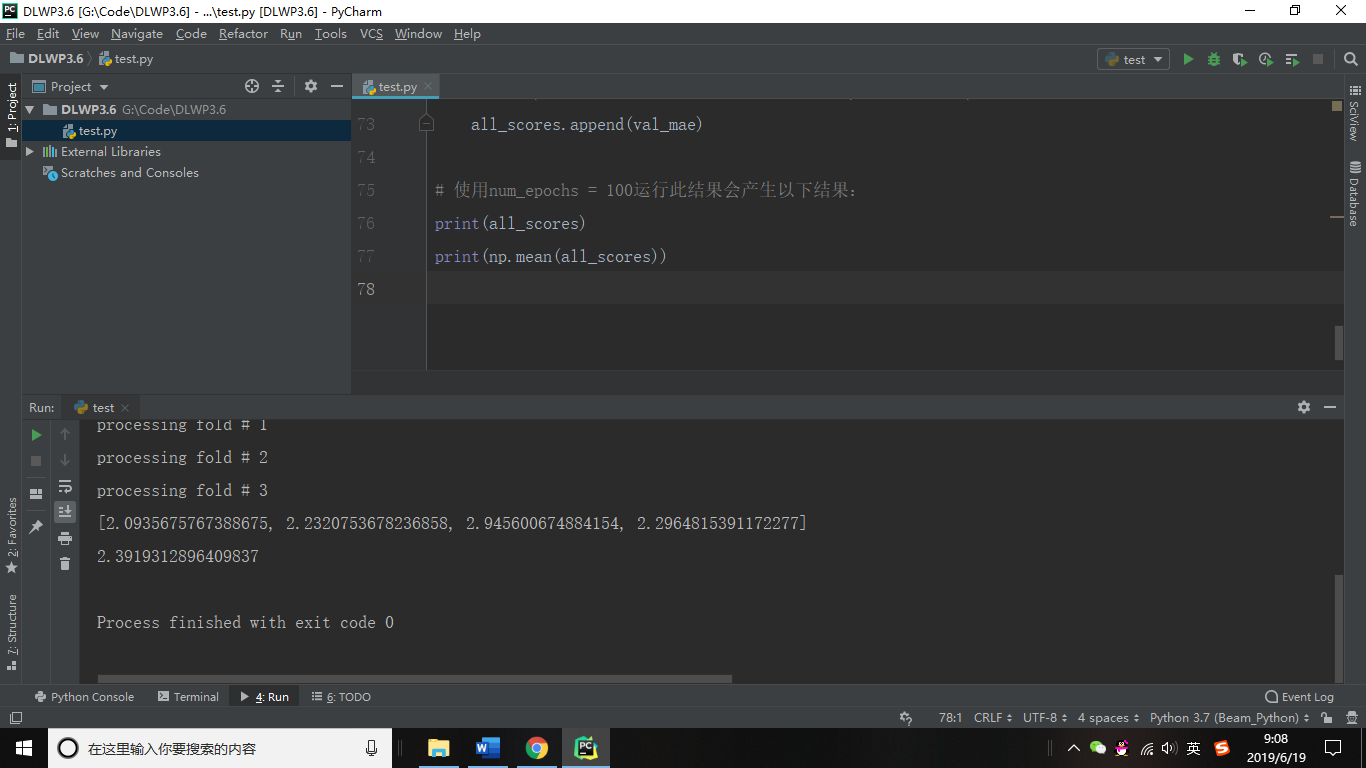
3）当输入数据中的要素具有不同范围的值时，应将每个要素作为预处理步骤单独进行缩放。

4）当数据很少时，使用K-fold验证是可靠评估模型的好方法。

5）当可用的训练数据很少时，最好使用一个隐藏层较少的小型网络（通常只有一个或两个），以避免严重的过度拟合。

运行结果：

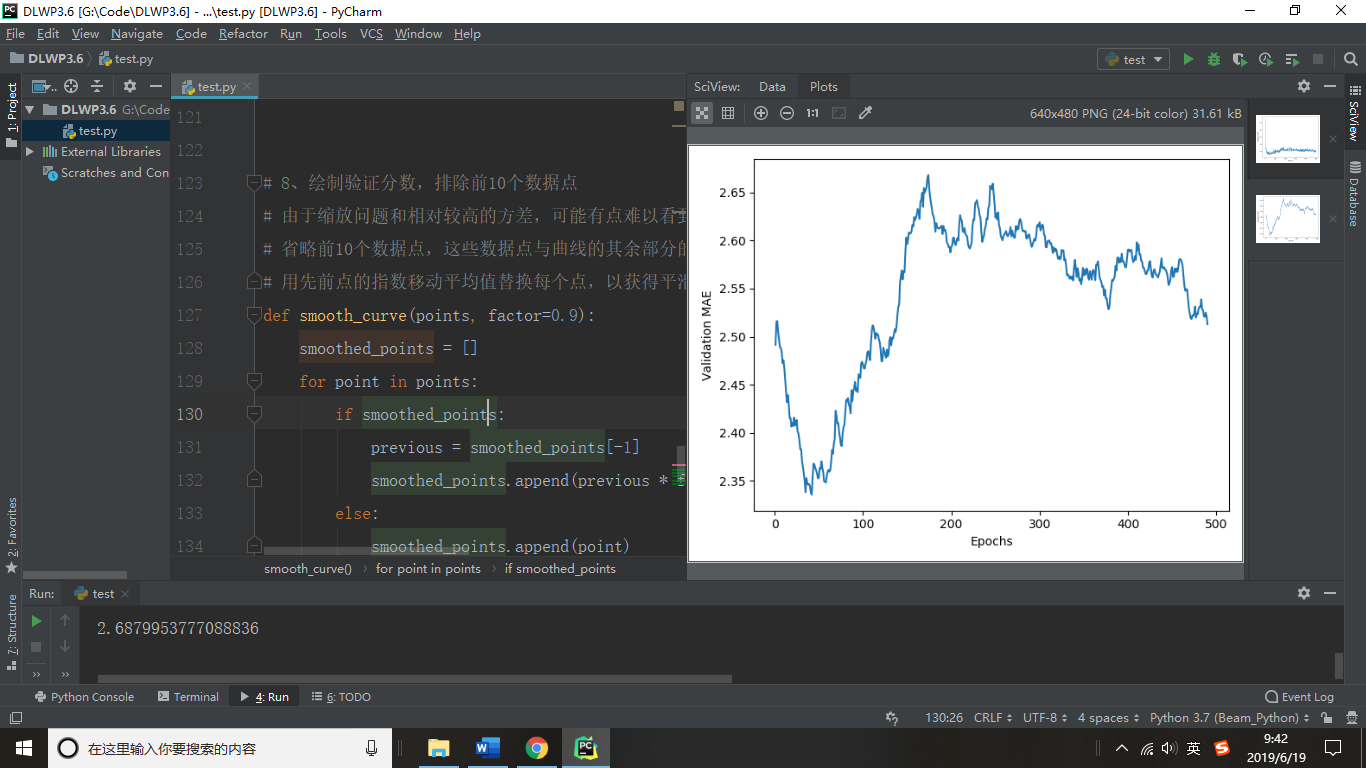
K-fold validation：



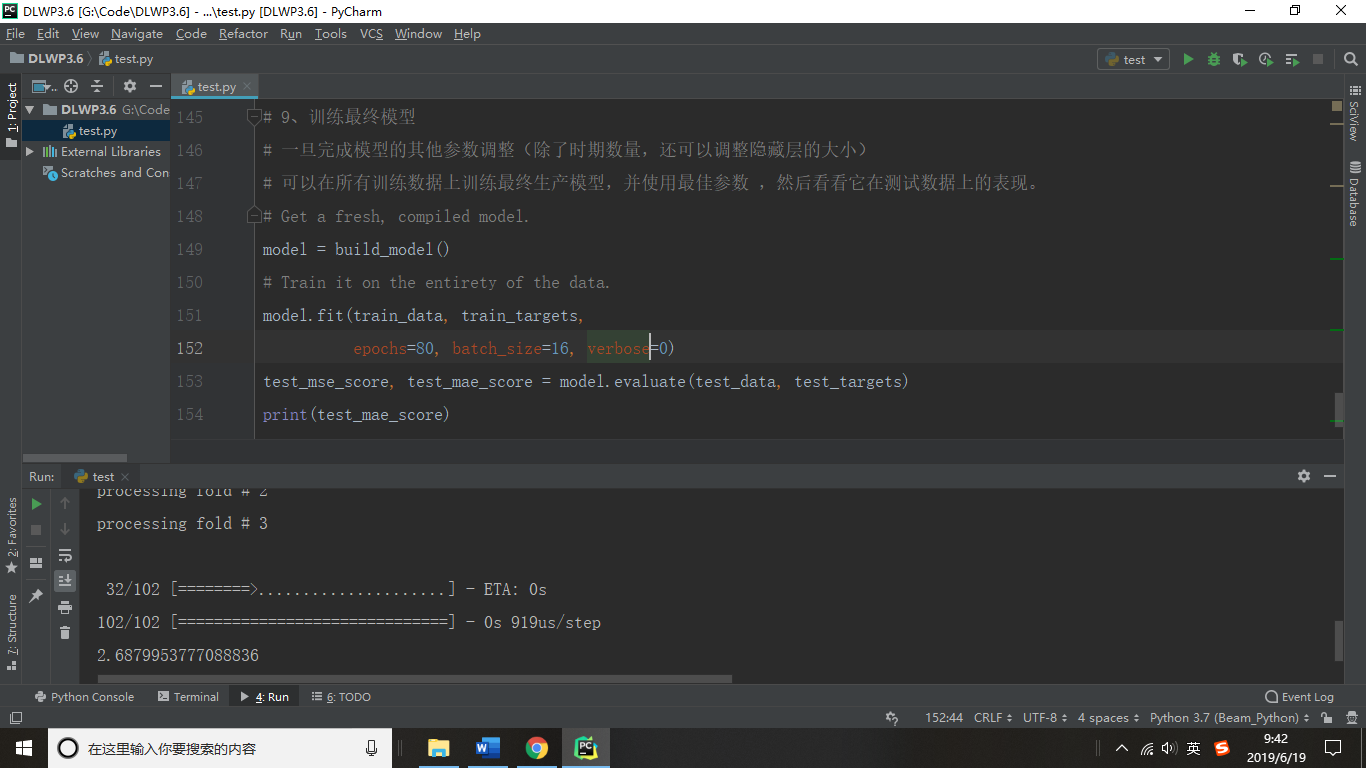
绘制验证分数：



绘制排除前10个数据点的验证分数 ：



训练最终模型，结果：



我们仍然偏离约$ 2,678

## 本章总结

1、目前为止，可以处理矢量数据中最常见的机器学习任务：二分分类，多类分类和标量回归。

2、通常需要在将原始数据输入神经网络之前对其进行预处理。

3、当数据具有不同范围的要素时，要在预处理过程中单独缩放每个要素。

4、随着训练的进行，神经网络最终开始过度拟合并在前所未见的数据上获得更差的结果。

5、如果没有太多的训练数据，使用只有一个或两个隐藏层的小型网络，避免严重过度拟合。

6、如果数据分为多个类别，如果使中间层太小，可能会导致信息瓶颈。

7、回归使用不同的损失函数和不同的评估指标而不是分类。

8、当您处理少量数据时，K-fold验证可以可靠地评估模型。

# Part1-4：Fundamentals of machine learning

本章包括：

1、超越分类和回归的机器学习形式

2、机器学习模型的正式评估程序

3、为深度学习准备数据

4、特征工程

5、解决过度拟合问题

6、接近机器学习问题的通用工作流程

## 一、机器学习的四个分支

机器学习算法通常分为四大类：

### 1、监督学习

监督学习主要包括分类和回归，但也有更多的外来变种，包括以下（有例子）：

1. 序列生成 - 给定图片，预测描述它的标题。 序列生成有时可以重新表述为一系列分类问题（例如重复预测序列中的单词或标记）。
2. 语法树预测 - 给定一个句子，将其分解预测为语法树。
3. 物体检测 - 给定图片，在图片内的某些对象周围绘制一个边界框。 这也可以表示为分类问题（给定许多候选边界框，对每个边界框的内容进行分类）或者作为联合分类和回归问题，其中边界框坐标是通过矢量回归预测。
4. 图像分割 - 给定图片，在特定对象上绘制像素级蒙版。

### 2、无监督学习

机器学习的这个分支包括在没有任何目标帮助的情况下发现输入数据的有趣变换，用于数据可视化，数据压缩或数据去噪，或者更好地理解手头数据中存在的相关性。

无监督学习是数据分析的基础，而且在尝试解决监督学习问题之前，通常是更好地理解数据集的必要步骤。维度降低和聚类是无人监督学习的众所周知的类别。

### 3、自主学习

自我监督学习是一种没有人工注释标签的监督学习，仍然涉及标签（因为学习必须由某些东西监督），但它们是从输入数据生成的，通常使用启发式算法。 例如，自动编码器。

### 4、强化学习

强化学习中，代理人会收到有关其环境的信息，并学会选择最大化某些奖励的行动。 例如，可以通过强化学习来训练“观察”视频游戏屏幕并输出游戏动作以最大化其得分的神经网络。

### 5、分类和回归词汇表

Sample or input（样本或输入）

Prediction or output（预测或输出）

Target（目标）

Prediction error or loss value（预测误差或损失值 - 衡量你的距离模型的预测和目标）

Classes（类 - 在分类问题中可供选择的一组可能标签。例如，在对猫和狗图片进行分类时，“dog”和“cat”是两个类）

Label（标签 - 分类问题中类注释的特定实例。例如，如果图片＃1234注释为包含类“dog”，则“dog”是图片＃1234的标签）

Ground-truth or annotations（实况或注释 - 数据集的所有目标，通常由人收集）

Binary classification（二分分类 - 每个输入样本应分为两个独有的类别）

Multiclass classification（多类分类 - 每个输入样本应分为两类以上：例如，对手写数字进行分类）

Multilabel classification（多标签分类 - 每个输入样本可以分配多个标签。例如，给定的图像可能包含猫和狗，并且应该使用“cat”标签和“dog”标签进行注释。每个图像的标签数量通常是可变的）

Scalar regression（标量回归 - 目标是连续标量值的任务。 预测房价就是一个很好的例子：不同的目标价格形成了一个连续的空间）

Vector regression（向量回归 - 目标是一组连续值的任务：例如，连续向量。对多个值（例如图像中的边界框的坐标）进行回归）

Mini-batch or batch（小批量或批量 - 由模型同时处理的一小组样品（通常在8到128之间）。 样本数通常是2的幂，以便于在GPU上进行内存分配。 训练时，使用小批量计算应用于模型权重的单个梯度下降更新。）

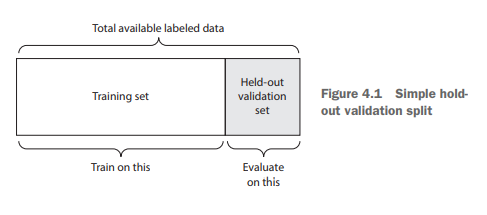
## 二、[评估机器学习模型](https://github.com/592595/DeepLearning/blob/master/4.2-validation-method.py)

### 1、训练、验证和测试集

三个经典的评估方法：

1. Hold-out validation（保留验证）

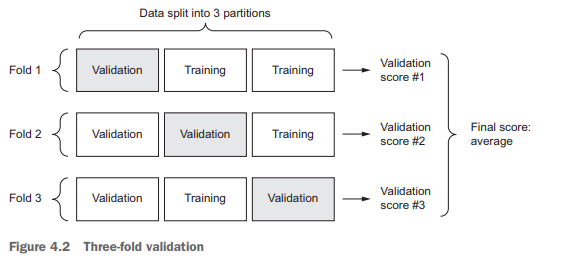
将一部分数据分开作为测试集。训练剩余数据，并在测试集上进行评估。为了防止信息泄漏，不应该根据测试集调整模型，因此还应该保留验证集。



2）K-fold cross-validation（交叉检验）

可以将数据拆分为相同大小的K个分区。 对于每个分区i，在剩余的K-1分区上训练模型，并在分区i上进行评估。最终得分是获得的K得分的平均值。

当模型的性能根据traintest分割显示出显着差异时，此方法很有用。 与Hold-out validation一样，此方法不会免除使用不同的验证集进行模型校准。



3）iterated K-fold validation with shuffling（带有改组的迭代K-fold验证）

### 2、要记住的事情

1）在将数据拆分为训练集和测试集之前，通常应该随机随机地重新调整数据。

2）时间的箭头 - 试图预测过去的未来（例如，明天的天气，股票走势等），不应该在拆分数据之前随机进行数据清洗，应始终确保测试集中的所有数据都位于训练集中的数据之后。

3）确保您的训练集和验证集不相交。

## 三、数据预处理，特征工程和特征学习

### 1、神经网络的数据预处理

数据预处理旨在使手头的原始数据更适合神经网络。 这包括矢量化，标准化，处理缺失值和特征提取。

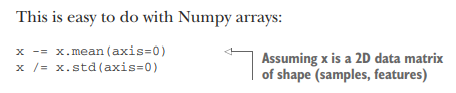
神经网络中的所有输入和目标必须是浮点数据的张量（或者，在特定情况下，是整数的张量）。 无论需要处理什么数据 - 声音，图像，文本 - 必须首先转变为张量，这一步称为数据矢量化。

一般来说，输入采用相对较大值的神经网络数据（例如，多位数整数，比网络权重所采用的初始值大得多）或异构数据（对于例如，一个特征在0-1范围内而另一个特征在100-200范围内的数据。这样做会触发大的梯度更新，从而阻止网络收敛。为了使网络更容易学习，数据应具有以下特征：

1. Take small values - 通常，大多数值应在0-1范围内。
2. Be homogenous - 也就是说，所有特征都应该在大致相同的范围内取值

此外，以下更严格的规范化实践很常见并且可以提供帮助，但并不总是必要的（例如，您没有在数字分类示例中执行此操作）：

1. 单独标准化每个特征，使其平均值为0。
2. 单独标准化每个特征以使标准偏差为1。



注意，如果你期望测试数据中缺少值，但网络已经过数据训练而没有任何缺失值，则网络将不会学会忽略缺失值！ 在这种情况下，应该人为地生成缺少条目的训练样本：多次复制一些训练样本，并删除测试数据中可能缺少的一些功能。

### 2、特征工程

特征工程是使用自己的数据知识和手头的机器学习算法的过程，通过在数据进行之前对数据应用硬编码（非学习）转换，使算法更好地工作进入模型。在许多情况下，期望机器学习模型能够从完全任意的数据中学习是不合理的。需要以一种使模型工作更容易的方式将数据呈现给模型。

## 四、[过度拟合和欠拟合](https://github.com/592595/DeepLearning/blob/master/4.4-overfitting-and-underfitting.py)

机器学习的根本问题是优化和泛化之间的紧张关系。优化是指调整模型以在训练数据（机器学习中的学习）上获得最佳性能的过程，而泛化是指训练模型对之前从未见过的数据的执行情况。

在训练开始时，优化和泛化是相关的：训练数据的损失越低，测试数据的损失越低。在对训练数据进行了一定数量的迭代之后，泛化停止了改进，验证指标停滞了然后开始降级：模型开始过度拟合。

 最好的解决方案是获得更多的训练数据。如果无法做到这一点，调整允许模型存储的信息量，或者为允许存储的信息添加约束。以这种方式对抗过度拟合的处理称为正则化。回顾一些最常见的正则化技术，并在实践中应用它们来改进前面的电影分类模型。

减少过度拟合的四种方法：**Getting more training data、Reducing the capacity of the network、Adding weight regularization、Adding dropout.**

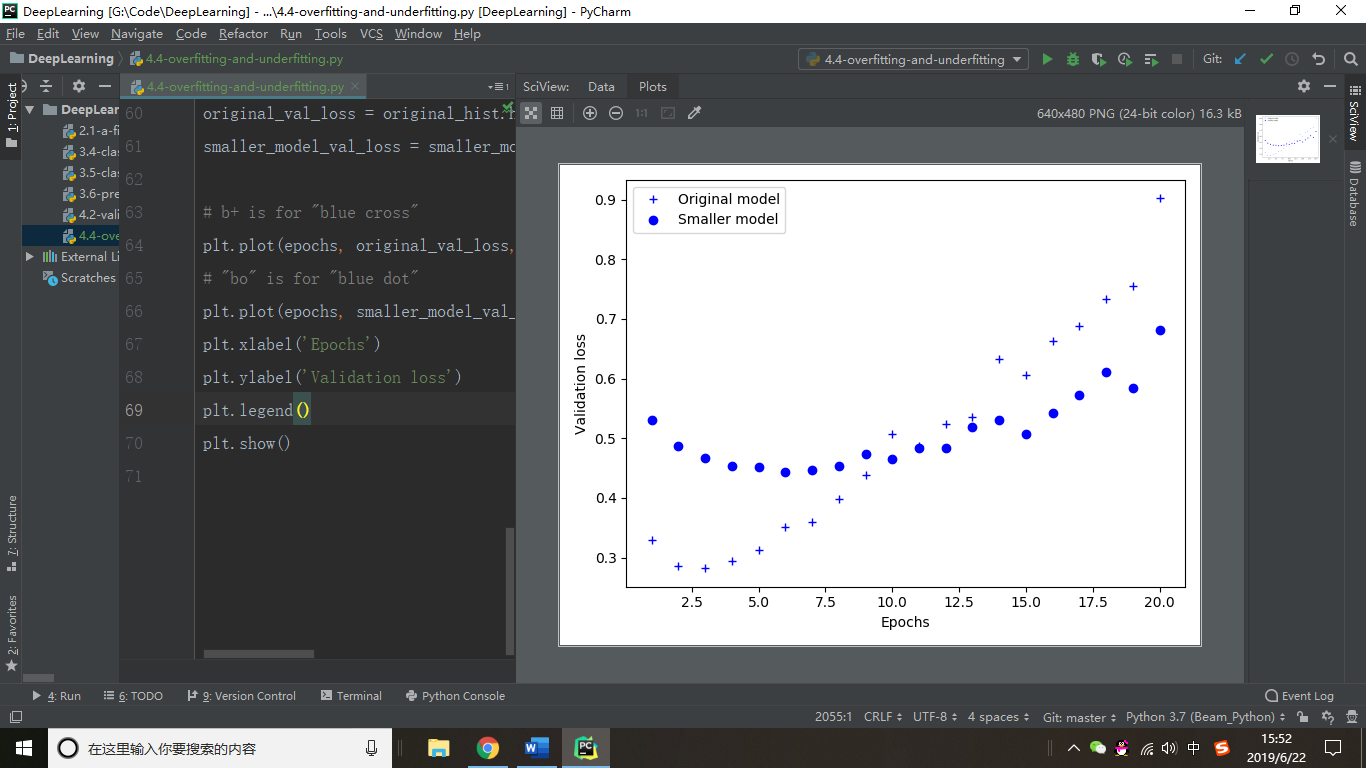
### 缩小网络的大小

防止过度拟合的最简单方法是减小模型的大小：模型中可学习参数的数量（由层数和每层单元数决定）。在深度学习中，模型中可学习参数的数量通常被称为模型的容量。具有更多参数的模型具有更多的记忆能力，因此可以容易地在训练样本和它们的目标之间学习完美的字典式映射 - 没有任何泛化能力的映射。始终牢记这一点：深度学习模型倾向于擅长拟合训练数据，但真正的挑战是泛化，而不是拟合。模型不应该缺乏记忆资源，在容量过大和容量不足之间存在折衷。

但是，没有确定的公式来确定每层的正确数量或正确的大小。必须评估一组不同的体系结构（当然不在测试集上，在验证集上），以便为数据找到正确的模型大小。查找适当模型大小的一般工作流程是从相对较少的图层和参数开始，并增加图层的大小或添加新图层，直到看到有关验证丢失的收益递减。

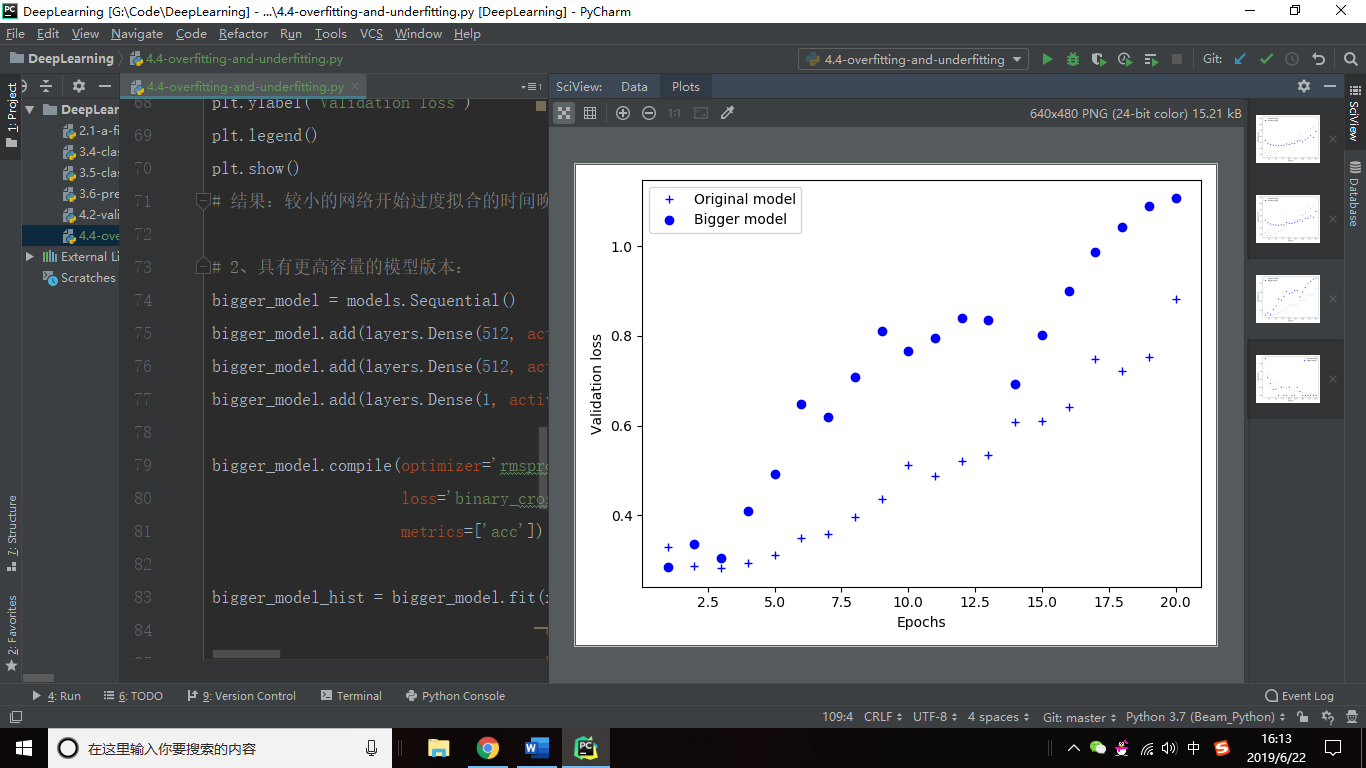
电影评论分类网络的例子试一试：

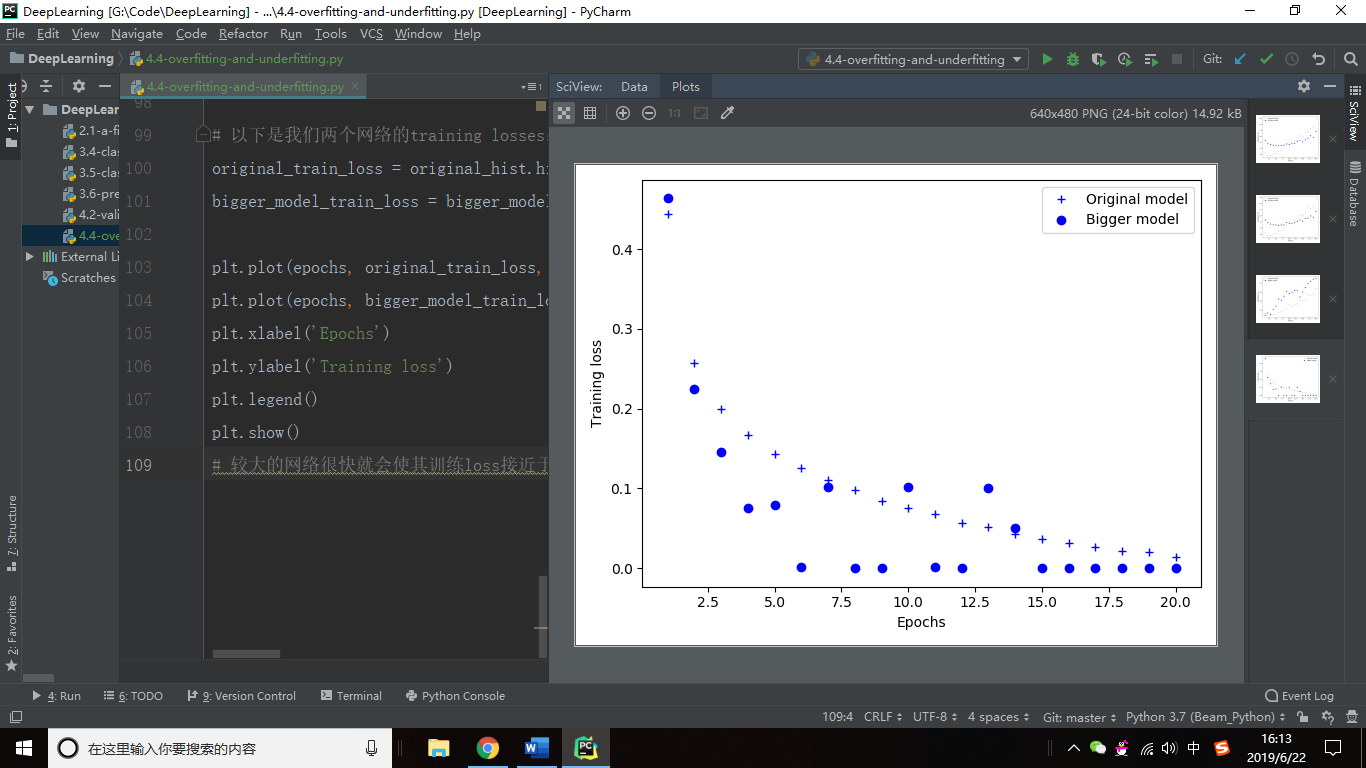
1）原始网络和较小网络的验证loss的比较：



结果：较小的网络开始过度拟合的时间晚于原网络（在6个时期之后而不是4个时期），并且一旦开始过度拟合，其性能就会降低得慢得多。

1. 具有更高容量的模型版本：





在一个epoch之后，更大的网络几乎立即开始过度拟合，并且更严重地过度拟合。它的验证loss也更加嘈杂。

较大的网络很快就会使其训练loss接近于零。网络容量越大，能够越快地对训练数据进行建模（导致训练loss低），但过度拟合的可能性越大（导致训练loss和验证loss之间的差异很大）。

### 加权正则化

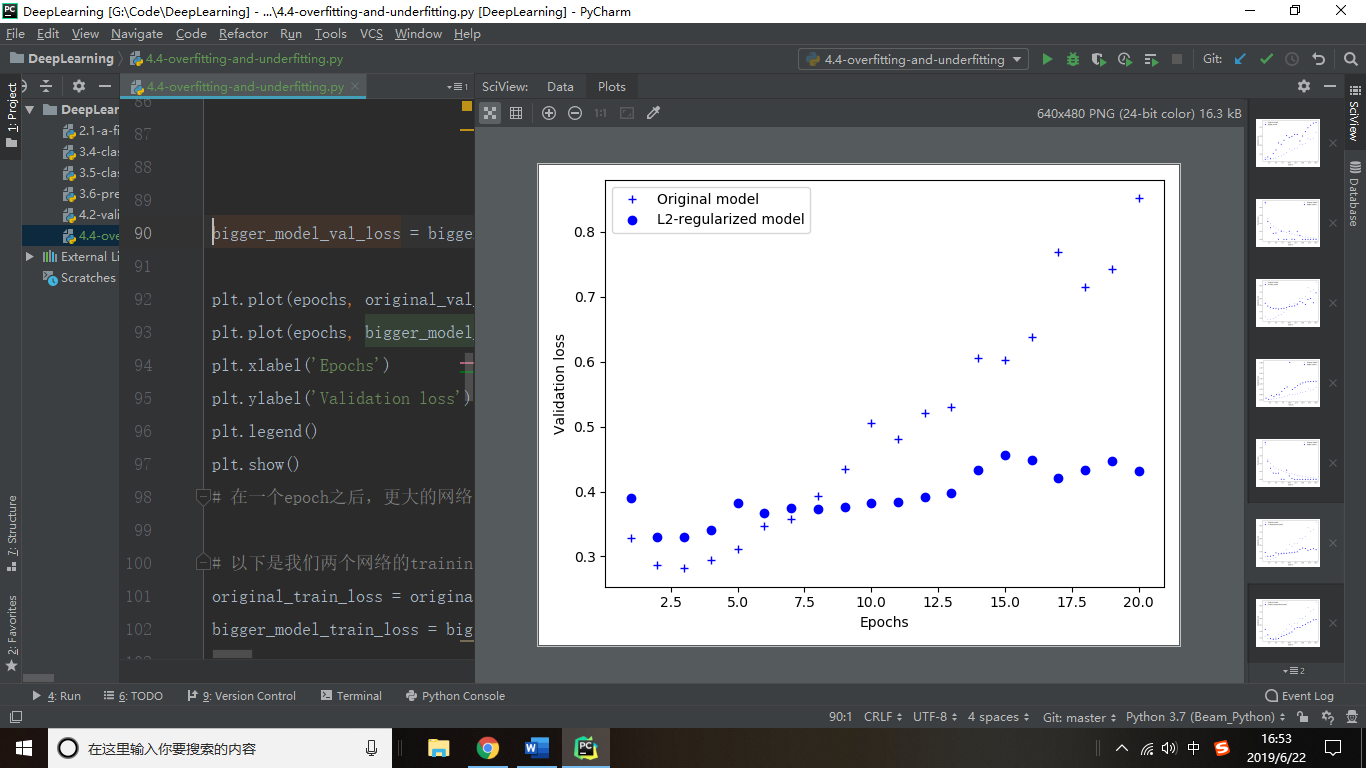
给定一些训练数据和网络架构，有多组权重值（多个模型）可以解释数据，而简单模型比复杂模型更不容易过度拟合。“简单模型”是一个具有较少参数的模型。因此，减轻过度拟合的常见方法是通过强制其权重仅采用较小的值来对网络的复杂性施加约束，这使得权重值的分布更“规则”。这被称为“权重正则化”，并且通过向网络的损失函数添加与具有大权重相关联的cost来完成。这个cost有两种：

L1正则化，其中所添加的成本与权重系数的绝对值成比例。

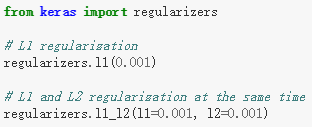
L2正则化，其中所添加的成本与权重系数的值的平方成比例。L2正则化在神经网络的背景下也称为权重衰减。

在Keras中，通过将权重正则化实例作为关键字参数传递给层来添加权重正则化。

以电影评论分类网络中添加L2权重正则化为例：



结果：具有L2正则化（点）的模型已经变得比参考模型（交叉）更能抵抗过度拟合，即使两个模型具有相同数量的参数。作为L2正规化的替代方案，您可以使用以下Keras重量正则化器之一：

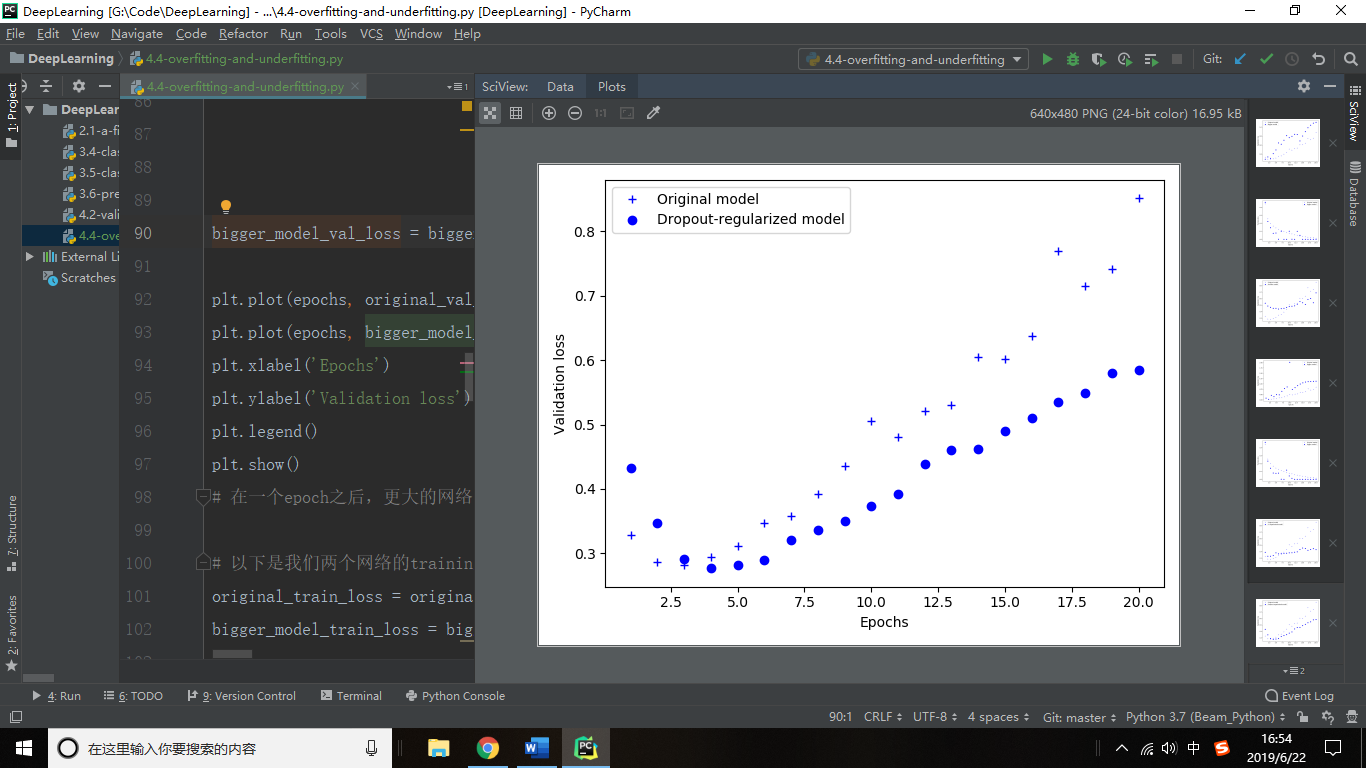


### 添加dropout

dropout应用于层，包括在训练期间随机“退出”（即设置为零）该层的多个输出特征。假设一个给定的层通常会在训练期间为给定的输入样本返回一个向量[0.2,0.5,1.3,0.8,1.1]; 在应用了丢失之后，该向量将具有随机分布的几个零条目，例如：[0,0.5,1.3,0,1.1]。

"dropout rate” 是被淘汰的特征的一部分; 它通常设置在0.2和0.5之间。在测试时，图层的输出值按照等于dropout rate的因子按比例缩小。

在IMDB网络中添加两个Dropout图层，看看它们在减少过度拟合方面做得如何：



## 五、机器学习的通用工作流程

### 1、定义问题并组装数据集

1）输入数据是什么？ 想要预测什么？

2) 面临什么类型的问题？ 是二元分类吗？ 多类分类？ 标量回归？ 矢量回归？ 多类，多标签分类？识别问题类型将指导我们去选择模型体系结构，损失函数等。

### 2、选择成功的衡量标准

对于balanced-classification问题，每个类别同样可能，接收器操作特性曲线（ROC AUC）下的准确度和面积是常见的度量标准。

对于class-imbalanced的问题，可以使用precision and recall。

对于ranking问题或multilabel classification分类，可以使用average precision。

或者定义自己的自定义指标，Kaggle（https://kaggle.com）浏览数据科学竞赛展示了广泛的问题和评估指标。

### 3、决定评估方案

三种常见的评估协议：

1. Maintaining a hold-out validation set：拥有大量数据时的方法。
2. Doing K-fold cross-validation：样本太少时的方法。
3. Doing iterated K-fold validation：当有少量数据可用时，执行高度精确的模型评估只需选择其中一个即可。 在大多数情况下，第一个将足够好。

### 4、准备数据

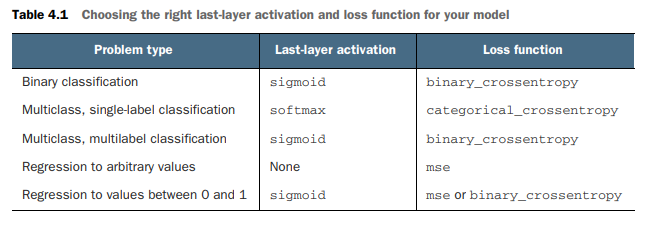
1. 数据应格式化为张量。
2. 这些张量所取的值通常应缩放为较小的值：例如，在[-1,1]范围或[0,1]范围内。
3. 如果不同的特征采用不同范围的值（异构数据），则应对数据进行标准化。
4. 可能需要进行一些特征工程，尤其是小数据问题。

一旦输入数据和目标数据的张量准备就绪，就可以开始训练模型。

### 5、开发一个比baseline更好的模型

需要做出三个关键选择来构建第一个工作模型：

1. Last-layer activation：例如，IMDB分类示例在最后一层使用了sigmoid; 回归示例没有使用任何最后一层激活; 等等。
2. Loss function：例如，IMDB示例使用binary\_crossentropy，回归示例使用mse，依此类推。
3. Optimization configuration：在大多数情况下，使用rmsprop及其默认学习速率是安全的。



### 6、扩大规模：开发一个过度拟合的模型

要弄清楚你需要多大的模型，你必须开发一个过度拟合的模型。方法：

1. Add layers.
2. Make the layers bigger.
3. Train for more epochs.

当看到模型在验证数据上的性能开始下降时，就已经实现了过度拟合。下一阶段是开始正规化和调整模型，以尽可能接近理想的模型，既不欠拟合也不过度拟合。

### 7、调整模型和超参数

方法：

1. Add dropout.
2. Try different architectures: add or remove layers.
3. Add L1 and/or L2 regularization.
4. Try different hyperparameters（例如每层的单位数或优化程序的学习率）以找到最佳配置
5. Optionally, iterate on feature engineering：添加新功能，或删除似乎没有提供信息的功能。

## 本章总结

1、定义手头上的问题和要训练的数据。收集这些数据，或者在需要时用标签对其进行注释。

2、选择如何衡量问题的成功。将监控验证数据的哪些指标？

3、确定评估方案：hold-out验证？K-fold验证？应该使用数据的哪个部分进行验证？

4、开发比基本baseline更好的第一个模型：具有统计能力的模型。

5、开发一个过度拟合的模型。

6、根据验证数据的性能调整模型并调整其超参数。很多机器学习研究倾向于只关注这一步，但要记住大局。

# Part2-5：Deep learning for computer vision

本章包括：

1、了解卷积神经网络（convnets）

2、使用数据增强来减轻过度拟合

3、使用预训练的convnet进行特征提取

4、微调预训练的信号

5、可视化回馈学习的内容以及他们如何做出分类决策

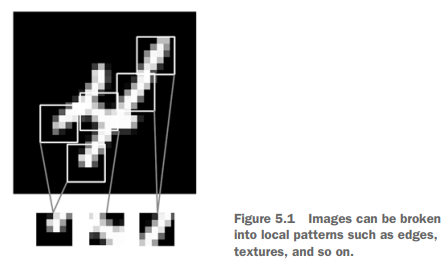
## 一、引言

[一个简单的convnet示例。](https://github.com/592595/DeepLearning/blob/master/5.1-introduction-to-convnets.py)

第二章中的密接网络的测试精度为97.8%，而convnet的测试精度为99.3%：我们将错误率降低了68%（相对）。为什么这个简单的convnet比一个紧密连接的模型工作得那么好呢？为了回答这个问题，让我们深入研究conv2d和maxpooling2d层的作用。

### 1、卷积操作

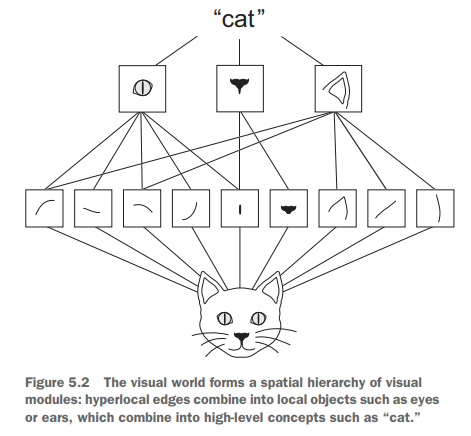
密集连接层和卷积层的根本区别在于：密集层在其输入特征空间中学习全局模式（例如，对于mnist数字，涉及所有像素的模式），而卷积层学习局部模式（参见图5.1）：对于图像，在输入。在前面的示例中，这些窗口都是3×3。



这个关键特性为convnets提供了两个有趣的属性：

1）他们学习的模式是翻译不变的---学习了这种模式之后，convnets可以在别的任何地方识别出来；这使得convnets在处理图像时的数据效率更高（因为视觉世界基本上是不变的）。

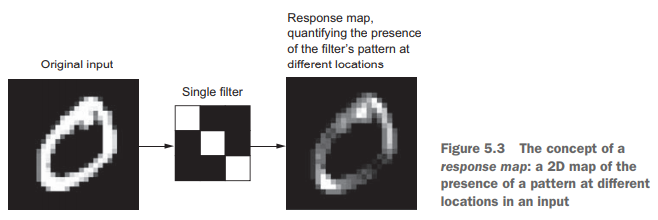
2）他们可以学习模式的空间层次结构（见图5.2）。 第一卷积层将学习诸如边缘的小局部图案，第二卷积层将学习由第一层的特征构成的较大图案，等等。 这使得convnets能够有效地学习越来越复杂和抽象的视觉概念（因为视觉世界从根本上讲是空间分层的）。



卷积作用于称为特征映射的三维张量上，它有两个空间轴（高度和宽度）和一个深度轴（也称为通道轴）。对于RGB图像，深度轴的尺寸为3，因为图像有三个颜色通道：红色、绿色和蓝色。对于黑白图片，像mnist数字一样，深度为1（灰度）。

卷积操作从其输入特征映射中提取补丁，并对所有这些补丁应用相同的转换，生成输出特征映射。这个输出特征映射仍然是一个三维张量：它有一个宽度和一个高度。它的深度可以是任意的，因为输出深度是图层的一个参数，并且该深度轴上的不同通道不再代表特定的颜色，就像在RGB输入中一样；相反，它们代表过滤器。

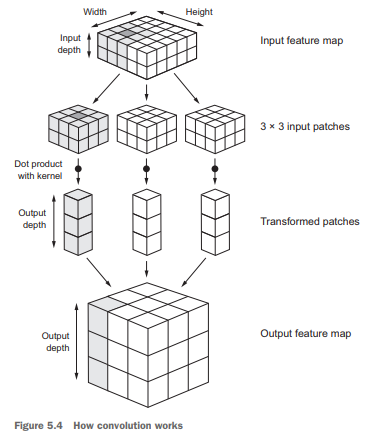
过滤器对输入数据的特定方面进行编码：例如，在高层，一个过滤器可以对“输入中存在一个面”的概念进行编码。在mnist示例中，第一个卷积层获取大小为（28、28、1）的特征映射并输出大小为（26、26、32）的特征映射：它在其输入上计算32个过滤器。这32个输出通道中的每个通道都包含一个26×26的网格值，这是输入端滤波器的响应图，指示该滤波器模式在输入端不同位置的响应（见图5.3）。这就是术语“特征映射”的含义：深度轴中的每个维度都是特征（或过滤器），而二维张量输出[：，：，n]是该过滤器对输入响应的二维空间映射。



卷积由两个关键参数定义：

1. 从输入中提取的补丁大小---通常为3×3或5×5
2. 输出特征映射的深度---由卷积计算的滤波器数量。该示例的开始深度为32，结束深度为64。

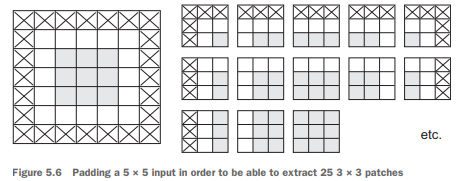
在keras conv2d层中，这些参数是传递给层的第一个参数：Conv2D(output\_depth, (window\_height, window\_width)) 卷积的工作原理是将这些尺寸为3×3或5×5的窗口滑动到三维输入特征映射上，在每个可能的位置停止，并提取周围特征的三维面片(shape (window\_height, window\_width, input\_depth)) 然后，所有这些向量在空间上重新组合成三维形状输出图(height,width, output\_depth) 输出特征映射中的每个空间位置对应于输入特征映射中的相同位置（例如，输出的右下角包含有关输入右下角的信息）。例如，使用3×3窗口时，矢量输出[i, j, :]来自3D补丁输入[i-1:i+1, j-1:j+1, :]。整个过程如图5.4所示。



注意，输出宽度和高度可能与输入宽度和高度可能因两个原因而不同：

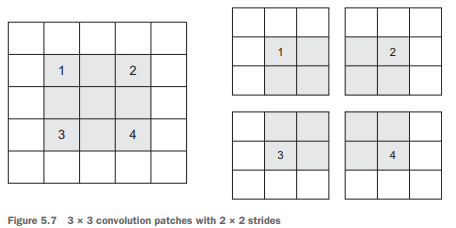
1. Border effects：可通过填充输入特征映射来抵消。
2. The use of strides

填充：如果要获得与输入空间尺寸相同的输出特征映射，可以使用填充。填充包括在输入功能图的每一侧添加适当数量的行和列，以便在每个输入图块周围放置中心卷积窗口。



在conv2d层中，padding可通过padding参数进行配置，padding参数采用两个值：“valid”，表示没有padding（只使用有效的窗口位置）；“same”（相同），表示“pad的方式使输出具有与输入相同的宽度和高度。”padding参数默认为“valid”。

影响输出大小的另一个因素是步幅的概念。迄今为止对卷积的描述假定卷积窗口的中心瓷砖都是连续的。但两个连续窗口之间的距离是卷积，称为步幅，默认为1。跨步卷积：跨步大于1的卷积。在图5.7中，可以看到通过3×3卷积在5×5输入（不带填充）上使用跨距2提取的补丁。



使用“跨距2”意味着特征地图的宽度和高度将被一个系数2（除了由边界效果引起的任何变化）所减小。跨步卷积在实践中很少使用，尽管对于某些类型的模型来说，跨步卷积可以派上用场；为了缩小功能图的样本，我们倾向于使用max pooling操作，而不是跨步，在第一个convnet示例中在action中看到了这一操作。

### 2、最大池操作

max pooling的作用：大幅缩小feature maps的样本，很像跨步卷积。最大池包括从输入功能图中提取窗口并输出每个通道的最大值。它在概念上类似于卷积，除了通过学习的线性变换（卷积核）转换局部补丁之外，它们通过硬编码的max张量操作进行转换。与卷积的一个大区别是，最大池通常使用2×2窗口和跨步2来完成，以便将特征映射的采样率降低2倍。卷积通常是用3×3个窗口完成的，没有跨距（跨距1）。使用降采样的原因是减少要处理的特征映射系数的数量，并通过使连续的卷积层查看越来越大的窗口（根据它们覆盖的原始输入的比例）来诱导空间滤波器层次。

注意，最大池并不是实现这种降采样的唯一方法。还可以在先前的卷积层中使用跨步。可以使用平均池而不是最大池，其中每个本地输入补丁都是通过在补丁上取每个通道的平均值而不是最大值来转换的，但是最大池比这些可选的解决方案工作得更好。

最合理的子采样策略是首先生成密集的特征图（通过无重叠卷积），然后查看小补丁上特征的最大激活，而不是查看输入窗口较稀疏（通过跨步卷积）或平均输入补丁，这可能会导致您错过或稀释特征-P重置信息。

此时，应该了解convnets特性图、卷积和max pooling的基本知识，并且知道如何构建一个小的convnet来解决诸如mnist digits分类之类的问题。

## 二、在小数据集上从头开始训练convnet

只有很少的数据来训练一个图像分类模型是一种常见的情况，在这个例子里，我们将把图像分类为“狗”或“猫”，在一个包含4000张猫和狗（2000只猫，2000只狗）图片的数据集中。我们将使用2000张图片进行训练，1000张用于验证，最后1000张用于测试。

在本节中，我们将回顾一个解决这个问题的基本策略：从头开始培训一个新模型，我们拥有的数据量很少。我们将从2000个训练样本上训练一个小的convnet开始，不进行任何正规化，为可以达到的目标设定一个基线。这将使我们的分类精度达到71%。在那一点上，我们的主要问题将是过度拟合。然后我们将介绍数据增强，这是一种减轻计算机视觉中过拟合的强大技术。通过利用数据扩充，我们将改进我们的网络，使其精度达到82%。

在下一节中，我们将回顾将深度学习应用于小数据集的两个更重要的技术：使用预先训练的网络进行特征提取（这将使我们的精度达到90%到93%），以及对预先训练的网络进行微调（这将使我们的最终精度达到95%）。总之，这三种策略——从头开始训练一个小模型，使用预先训练的模型进行特征提取，以及微调预先训练的模型——将构成未来解决使用小数据集进行计算机视觉的问题的工具箱。

### 深度学习对小数据问题的相关性

有时我们会觉得，只有在有大量数据可用时，深度学习才有效。这在一定程度上是一个有效的观点：深度学习的一个基本特征是，它能够自己在训练数据中发现有趣的特性，而不需要任何手动特性工程，并且只有在有大量训练示例的情况下才能实现。这对于输入样本是高维的问题尤其适用，比如图像。

然而，相对于您试图训练的网络的大小和深度而言构成“大量”样本的因素是相对的。用几十个样本训练一个convnet来解决一个复杂的问题是不可能的，但是如果模型很小并且很好地规范化，并且任务很简单，那么几百个样本就可能足够了。由于convnet学习了局部的、翻译不变的特征，因此它们在感知问题上具有很高的数据效率。从零开始在一个非常小的图像数据集上训练convnet仍然会产生合理的结果，尽管相对缺乏数据，但不需要任何定制的特性工程。

更重要的是，深度学习模型本质上是高度可重用的：例如，可以采用在大型数据集上训练的图像分类或语音到文本模型，在显著不同的问题上重用它，只需进行微小的更改。具体来说，在计算机视觉的情况下，许多预先训练的模型（通常在ImageNet数据集上训练）现在可以公开下载，并且可以用于从非常少的数据中引导强大的视觉模型。这就是我们将在下一节中要做的。现在，从掌握数据开始。

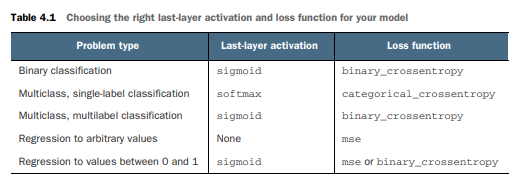
### 下载数据

<https://www.kaggle.com/c/dogs-vs-cats/data>

下载并解压缩后，我们将创建一个包含三个子集的新数据集：训练集每个类有1000个样本，一个验证集每个类有500个样本，最后一个测试集每个类有500个样本。原始数据集包含25000张狗和猫的图像（每类12500张）。每个类的样本数是相同的：这是一个平衡的二分分类问题，这意味着分类的准确性将是成功的适当度量。

### 构建网络

我们的convnet将是一个交替的conv2d（带有relu激活）和maxpooling2d层的堆栈。对于编译步骤，我们将像往常一样使用RMSprop优化器，loss的选择：binary crossentropy



### 数据预处理

在将数据输入我们的网络之前，应该将数据格式化为适当的预处理浮点张量。目前，我们的数据以jpeg文件的形式存储在驱动器上，因此将其导入网络的步骤大致如下：

1. 阅读图片文件。
2. 将jpeg内容解码为像素的RBG网格。
3. 把它们转换成浮点张量。
4. 将像素值（介于0和255之间）重新调整为[0，1]间隔（神经网络擅长处理较小的输入值）。

keras有一个带有图像处理助手工具的模块，位于keras.preprocessing.image。特别的是，它包含了ImageDataGenerator类，它允许快速设置python生成器，该生成器可以自动将磁盘上的图像文件转换成批预处理张量。这是我们将在这里使用的。

然后使用fit\_generator的方法实现生成器将模型与数据相拟合。

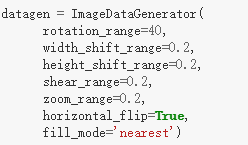
在训练后始终保存我们的模型。

用训练和验证数据来绘制模型在训练期间的loss 和accuracy。

由于训练样本相对较少，出现了过度拟合的现象，采用数据增强的方法来缓解过度拟合。

### 使用数据增强

数据增强采用从现有训练样本生成更多训练数据的方法，通过一些随机转换“增强”样本，生成可信的图像。我们的目标是，在培训时，我们的模型不会看到完全相同的图片两次。这有助于模型暴露于数据的更多方面，并更好地进行归纳。在Keras中，这可以通过配置要对ImageDataGenerator实例读取的图像执行的多个随机转换来完成。一个例子：



1. rotation\_range是一个以度（0-180）为单位的值，在这个范围内可以随机旋转图片。
2. width\_shift和height\_shift 是一个范围（作为总宽度或总高度的一部分），在这个范围内可以随机地垂直或水平地翻译图片。
3. shear\_range 是指随机应用剪切变换。
4. zoom\_range 用于在图片内随机缩放。
5. horizontal\_flip是指将一半图像随机水平翻转，这与没有水平不对称假设（如真实图像）相关。
6. fill\_mode 是用于填充新创建的像素的策略，这些像素可以在旋转或宽度/高度移动后出现。

## 三、使用预先训练的convnet

对小图像数据集进行深入学习的一种常见且高效的方法是利用预先培训过的网络。预先训练的网络只是一个保存的网络，以前是在一个大型数据集上训练的，通常是在一个大型图像分类任务上。

在我们的例子中，我们将考虑在ImageNet数据集上训练的大型convnet（140万个标记图像和1000个不同的类）。Imagenet包含许多动物类别，包括不同种类的猫和狗，因此我们可以期望在猫和狗的分类问题上表现出色。

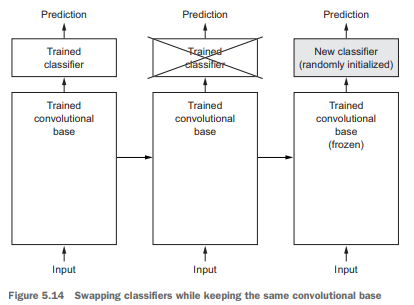
我们将使用由Karen Simonyan和Andrew Zisserman于2014年开发的VGG16架构，这是一种简单且广泛使用的用于Imagenet的convnet架构。

利用预先培训的网络有两种方法：特征提取和微调。我们将涵盖这两个方面。让我们从特征提取开始。

### 特征提取

特征提取包括使用以前网络学习的表示从新样本中提取有趣的特征。然后，这些特性通过一个新的分类器运行，这个分类器是从零开始训练的。

正如我们前面做的，用于图像分类的convnets由两部分组成：它们从一系列池和卷积层开始，最后是一个紧密连接的分类器。第一部分是模型的“卷积基”。在convnets里，“特征提取”将简单地包括获取先前训练的网络的卷积基，通过它运行新的数据，并在输出的顶部训练一个新的分类器。



为什么只重用卷积基？我们也可以重用密接分类器吗？一般来说，最好不要这样做。另外，如果我们新数据集与训练原始模型的数据集有很大不同，那么最好只使用模型的前几层进行特征提取，而不是使用整个卷积基。

在我们的例子中，由于imagenet类集确实包含多个dog和cat类，因此重用原始模型的紧密连接层中包含的信息可能会有所帮助。然而，我们将选择不这样做，以涵盖更一般的情况，即新问题的类集不与原始模型的类集重叠。我们通过使用在ImageNet上训练的VGG16网络的卷积基，从猫和狗的图像中提取有趣的特征，然后在这些特征的基础上训练猫和狗的分类器，来实践这一点。VGG16型号，以及其他型号，都预装了Keras。可以从keras.applications模块导入它。以下是作为keras.applications的一部分提供的图像分类模型列表（所有对ImageNet数据集进行过预培训）。

* Xception
* InceptionV3
* ResNet50
* VGG16
* VGG19
* MobileNet

我们通过使用在ImageNet上训练的VGG16网络的卷积基，从猫和狗的图像中提取特征，然后在这些特征的基础上训练猫和狗的分类器，来实践这一点。步骤：

1. 实例化vg16模型
2. 特征提取的两种方法：

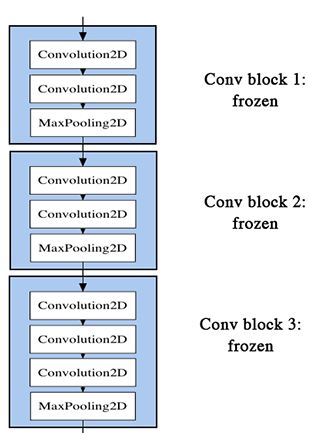
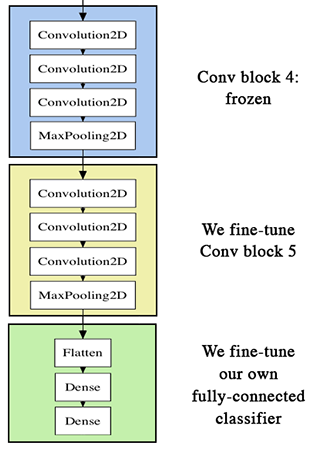
方法一：在数据集上运行卷积基，将其输出记录到磁盘上的一个numpy数组中，然后使用这个日期作为一个独立的、紧密连接的分类器的输入，类似于本书第一章中所看到的。这个解决方案运行起来非常快速和便宜，因为它只需要对每个输入图像运行一次卷积基，而卷积基是迄今为止管道中最昂贵的部分。然而这种技术根本不允许我们利用数据扩充。

方法二：扩展我们的模型（conv-），在顶部添加密集层，并在输入数据上端到端地运行整个模型。这允许我们使用数据增强，因为每次模型看到每个输入图像都要经过卷积基。然而这种技术要比第一种技术代价更大。只能在GPU上运行。

这里因为模型的行为与层类似，所以可以向顺序模型添加模型（如conv\_库），就像添加层一样。在我们编译和训练模型之前，要做的一件非常重要的事情是冻结卷积基。”“冻结”一层或一组层意味着在训练过程中防止权重得到更新。如果我们不这样做，那么之前由卷积基学习的表示将在训练期间被修改。由于顶部的密集层是随机初始化的，因此会通过网络传播非常大的权重更新，从而有效地破坏先前学习的表示。在keras中，通过将网络的可训练属性设置为false来冻结网络：conv\_base.trainable = False。使用此设置，将只训练我们添加的两个密集层的权重。总共有四个权重张量：每层两个（主权重矩阵和偏移向量）。注意，为了使这些更改生效，我们必须首先编译模型。如果在编译之后修改了权重可训练性，那么应该重新编译模型，否则这些更改将被忽略。现在，我们可以开始培训我们的模型，使用前面示例中使用的相同的数据扩充配置。

### 微调

另一种广泛使用的模型重用技术（与特征提取互补）是微调。微调包括解冻用于特征提取的冻结模型库的一些顶层，并联合训练模型的新添加部分（在我们的例子中，是完全连接的分类器）和这些顶层。这被称为“微调”，因为它稍微调整了被重用模型的更抽象的表示，以便使它们与手头的问题更相关。

我们之前已经说过，为了能够在顶部训练一个随机初始化的分类器，有必要冻结VGG16的卷积基。出于同样的原因，只有在对顶部分类器进行了训练之后，才能对卷积基的顶层进行微调。如果分类没有经过训练，那么在训练过程中通过网络传播的错误信号将太大，并且先前由被微调的层所学习的表示将被破坏。因此，对网络进行微调的步骤如下：

1）将我们自定义网络添加到已训练的基础网络之上。

2）冻结基础网络。

3）训练我们添加的部件。

4）解冻基础网络中的某些层。

5）共同训练这些层和添加的零件。

在进行特征提取时，我们已经完成了前3个步骤。让我们继续第四步：我们将解冻conv\_基地，然后冻结里面的各个层。

卷积基中的早期层编码更通用、可重用的特性，而更高层编码更专业的特性。对更专业的特性进行微调更有用，因为这些特性是需要针对新问题重新调整用途的特性。在微调较低层时，会有快速降低的回报。

我们训练的参数越多，就越有过度拟合的风险。卷积基有1500万个参数，因此尝试在我们的小数据集上训练它是有风险的。

因此，在我们的情况下，只对卷积基中的前2到3层进行微调是一个很好的策略。我们将对最后3个卷积层进行微调，这意味着在Block4\_池之前的所有层都应冻结，并且可以训练Block5\_Conv1、Block5\_Conv2和Block5\_Conv3层。

微调过程如下：

1）让我们来设置它，从上一个例子中我们停止的地方开始

2）微调我们的网络。我们将使用rmsprop优化器，使用非常低的学习率来实现这一点。使用低学习率的原因是，我们希望将所做的修改的大小限制为我们正在微调的3个层的表示。太大的更新可能会损害这些表示。

3）根据测试数据最终评估此模型

这里我们得到了97%的测试精度。在最初围绕这个数据集的竞争中，这将是最重要的结果之一。然而，利用现代的深度学习技术，我们仅使用了一小部分可用的培训数据（约10%）就取得了这一结果。与2000个样本相比，能够训练20000个样本有很大的区别！

使曲线更易读的方法，我们可以用这些量的指数移动平均值来替换每一个损失和精度，从而使它们更平滑。下面是一个简单的实用程序函数：



### 总结

1）convnets是计算机视觉任务的最佳机器学习模型。甚至可以在一个非常小的数据集上从头开始训练一个数据集，结果也不错。

2）在小数据集上，过度拟合将是主要问题。在处理图像数据时，数据增强是一种有效的克服过度拟合的方法。

3）通过特征提取，很容易在新数据集上重用现有的convnet。这对于处理小图像数据集是一种非常有价值的技术。

4）作为对特征提取的补充，我们可以使用微调，它可以适应一个新的问题，一些现有模型以前学习的表示。这会进一步提高性能。

现在，我们有了一套处理图像分类问题的可靠工具，特别是处理小数据集的工具。

## 四、可视化convnets学习的内容

人们常说，深度学习模型是“黑匣子”，学习表示很难以人类可读的形式提取和呈现。虽然对于某些类型的深度学习模型，这是部分正确的，但对于convnets，这绝对不是正确的。convnets学习的表示非常容易可视化，这在很大程度上是因为它们是可视化概念的表示。我们将涵盖以下三方面：

1）可视化中间convnet输出（“中间激活”）。这有助于理解连续的convnet层如何转换其输入，并首先了解各个convnet过滤器的含义。

2）可视化convnets筛选器。这有助于准确理解convnet中每个过滤器所能接受的视觉模式或概念。

3）在图像中可视化类激活的热图。这有助于理解标识为属于给定类的图像的哪个部分，从而允许在图像中定位对象。

对于第一种方法——激活可视化——我们将使用两节前从零开始训练的小convnet来解决猫和狗的分类问题。对于接下来的两种方法，我们将使用上一节中介绍的vg16模型。

### 1、Visualizing intermediate activations

可视化中间激活包括显示由网络中各种卷积和池层输出的特征图，给定一个特定的输入（一个层的输出通常称为其“激活”，即激活函数的输出）。这就为输入如何分解为网络学习的不同过滤器提供了一个视图。我们要可视化的这些功能图有三个维度：宽度、高度和深度（通道）。每个通道编码相对独立的特征，因此可视化这些特征图的正确方法是将每个通道的内容作为二维图像单独标绘。

这里需要注意的几个重要事项：

1）第一层是各种边缘检测器的集合。在那个阶段，激活仍然保留了最初图片中的几乎所有信息。

2）随着我们向上走，激活变得越来越抽象，视觉解释能力也越来越差。他们开始编码更高级的概念，如“猫耳”或“猫眼”。更高级别的演示文稿包含的关于图像视觉内容的信息越来越少，与图像类别相关的信息也越来越多。

3） 激活的稀疏性随着层的深度而增加：在第一层中，所有的过滤器都由输入图像激活，但在下面的层中，越来越多的过滤器是空白的。这意味着在输入图像中找不到由过滤器编码的模式。

深层神经网络所学习的表示的一个非常重要的普遍特征：一个层所提取的特征随着层的深度越来越抽象。更高层的激活会带来越来越少的关于所看到的特定输入的信息，以及越来越多的关于目标的信息（在我们的例子中，是图像的类别：猫或狗）。深度神经网络有效地充当信息蒸馏管道，原始数据进入（在我们的例子中，是RBG图片），并反复转换，以便过滤出不相关的信息（例如图像的特定视觉外观），而有用的信息被放大和细化（例如图像的类别）。

这类似于人类和动物感知世界的方式：观察一个场景几秒钟后，人类可以记住其中存在的抽象对象（如自行车、树），但无法记住这些对象的具体外观。事实上，如果你现在试图从脑海中画出一辆普通的自行车，很有可能你根本就不能把它画对，即使你一生中见过成千上万辆自行车。你的大脑已经学会了完全抽象它的视觉输入，把它转换成高层次的视觉概念，同时完全过滤掉不相关的视觉细节，这使得我们很难记住周围事物的实际外观。

### 2、Visualizing convnet filters

检查convnets学习的过滤器的另一个简单方法是显示每个过滤器要响应的可视模式。这可以通过输入空间中的梯度上升来实现：将梯度下降应用于convnet输入图像的值，以便从空白输入图像开始最大化特定过滤器的响应。所产生的输入图像将是所选滤波器最大响应的图像。这个过程很简单：我们将建立一个损失函数，使给定卷积层中给定滤波器的值最大化，然后我们将使用随机梯度下降来调整输入图像的值，从而使这个激活值最大化。例如，在VGG16网络的“block3\_conv1”层中激活过滤器0时会丢失，在ImageNet上进行了预训练。

这些过滤器可视化告诉我们很多关于convnet层如何看待世界的信息：convnet中的每一层都只是学习一组过滤器，这样它们的输入就可以表示为过滤器的组合。这类似于傅立叶变换如何将信号分解成一组余弦函数。随着我们在模型中的提升，这些convnet过滤器组中的过滤器变得越来越复杂和精细：

1）模型中第一层的过滤器（block1\_conv1）编码简单的方向性边和颜色（在某些情况下为彩色边）。

2）Block2\_Conv1的过滤器对由边和颜色组合而成的简单纹理进行编码。

3）更高层的过滤器开始类似于自然图像中的纹理：羽毛、眼睛、树叶等。

### 3、Visualizing heatmaps of class activation

我们将介绍另一种可视化技术，这种技术有助于理解给定图像的哪些部分引导其进行最终的分类决策。这有助于“调试”回转网的决策过程，特别是在分类错误的情况下。它还允许我们在图像中查找特定对象。

这种一般类别的技术称为“类激活图”（CAM）可视化，并且包括在输入图像上产生“类激活”的热图。 “类激活”热图是与特定输出类相关联的分数的2D网格，针对任何输入图像中的每个位置计算，指示每个位置相对于所考虑的类的重要程度。例如，如果将图像输入我们的“猫与狗”之一，则类激活图可视化允许我们为类“猫”生成热图，指示图像中猫的不同部分是如何的，同样如此对于“狗”类，表示图像的狗状不同部分。

我们将使用的具体实现是Grad-CAM中描述的实现：你为什么这么说？ 深度网络中基于梯度的本地化的可视化解释。 它非常简单：它包括在给定输入图像的情况下获取卷积层的输出特征图，并通过类相对于通道的梯度对该特征图中的每个通道进行加权。 直觉上，理解这一技巧的一种方法是，我们通过“每个通道对于类别的重要程度”来加权“输入图像激活不同通道的强度”的空间图，从而输入图像激活类的强烈程度”的空间图。

此可视化技术回答了两个重要问题：

1）为什么网络认为这个图像包含非洲象？

2）非洲大象在哪里？

特别值得注意的是，大象幼崽的耳朵被强烈激活：这可能是网络如何区分非洲和印度大象的区别。

## 本章总结

1、convnets是解决视觉分类问题的最佳工具。

2、convnets通过学习模块化模式和概念的层次结构来表示视觉世界。

3、他们学习的表示很容易检查-convnets与黑盒相反！

4、现在能够从头开始培训自己的convnet，以解决图像分类问题。

5、了解如何使用可视化数据增强来克服过度拟合。

6、知道如何使用预训练的convnet进行特征提取和微调。

7、可以生成convnets学习的过滤器的可视化，以及类活动的heatmap。

# Part2-6：Deep learning for text and sequences

本章包括：

1. 将文本数据预处理成有用的表示形式
2. 使用循环神经网络
3. 使用1d convnets进行序列处理

本章探讨了可以处理文本（理解为单词或字符序列），时间序列和序列数据的深度学习模型。用于序列处理的两种基本深度学习算法是递归神经网络和1D convnets，这是我们在前面章节中介绍的2D网络的一维版本。我们将在本章中讨论这两种方法

这些算法的应用包括以下内容：

* 1. 文档分类和时间序列分类，例如标识文章主题或书籍作者
  2. 时间序列比较，例如估计两个文档或两个股票行情的密切关系
  3. 序列到序列学习，例如将英语句子解码为法语
  4. 情绪分析，例如将推文或电影评论的情绪分类为正面或负面
  5. 根据最近的天气数据预测时间序列，例如预测某个地点的未来天气

本章的示例主要关注两个任务：IMDB数据集的情感分析，我们在本书前面讨论过的任务，以及温度预测。但是，针对这两项任务所展示的技术与刚刚列出的所有应用程序相关，还有更多。

## 一、使用文本数据

文本是最常见的序列数据形式之一。 它可以被理解为一系列字符或一系列单词，但最常见的是在单词层面上工作。 以下部分介绍的深度学习序列处理模型可以使用文本来产生自然语言理解的基本形式，足以应用于文档分类，情感分析，作者识别，甚至问答（QA）（在 受约束的上下文）。 当然，在本章中请记住，这些深度学习模型都不能真正理解人类意义上的文本; 相反，这些模型可以映射书面语言的统计结构，这足以解决许多简单的文本任务。 自然语言处理的深度学习是应用于**单词，句子和段落**的模式识别，其方式与计算机大致相同

视觉是应用于**像素**的模式识别。

与所有其他神经网络一样，深度学习模型不会将输入原始文本作为输入：它们仅适用于数字张量。 向量化文本是将文本转换为数字张量的过程。 这可以通过多种方式完成：

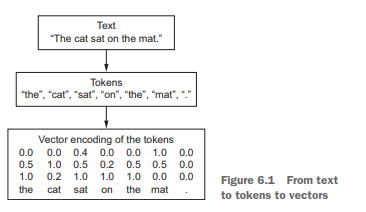
 1）将文本分段为单词，并将每个单词转换为向量。

2）将文本分段为字符，并将每个字符转换为矢量。

 3）提取n-gram单词或字符，并将每个n-gram转换为向量.N-gram是多个连续单词或字符的重叠组。

总的来说，可以分解文本（单词、字符或n-gram）的不同单元称为标记，而将文本分解为此类标记称为标记化技术。所有文本矢量化过程都包括应用一些标记化技术方案，然后将数字向量与生成的标记关联起来。这些矢量被压缩成序列张量，被送入深度神经网络。

有多种方法可以将向量与标记关联。在本节中，将介绍两种主要的方法：一种是**标记的热编码**，另一种是**标记嵌入**（通常只用于单词，称为单词嵌入）。本节的其余部分将解释这些技术，并说明如何使用它们将原始文本转换为可以发送到Keras网络的Numpy张量。



理解n-grams和 bag-of-words。由于bag-of-words不是一种保留顺序的标记化技术方法（生成的标记被理解为一个集合，而不是一个序列，并且句子的一般结构丢失），因此它倾向于用于浅语言处理模型，而不是用于深度学习模型。

提取n-grams是特征工程的一种形式，而深度学习则摒弃了这种刚性、脆弱的方法，取而代之的是层次特征学习。本章后面介绍的一维convnets和递归神经网络，通过观察连续的单词或字符序列，能够学习单词和字符组的表示，而无需明确地了解这些组的存在。但是当使用轻量级，浅层文本处理模型（如逻辑回归和随机森林）时，n-gram是一种强大的，不可避免的特征工程工具。

### 1、One-hot encoding of words and characters

One-hot encoding是将标记转换为向量的最常见、最基本的方法。可以在单词级别进行One-hot encoding，也可以在字符级别进行。两个示例：

1. 单词级别进行热编码：
2. 字符级别进行热编码：

注意：Keras具有内置实用程序，用于从原始文本数据开始在单词级别或字符级别执行单热编码文本。这是我们应该实际使用的内容，因为它将处理许多重要功能，例如从字符串中删除特殊字符，或者只接受数据集中最常见的N个最常用的单词（避免处理的常见限制） 非常大的输入向量空间）。例子：

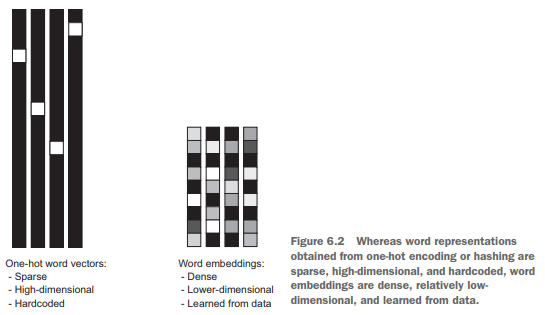
1. 使用Keras进行字级单热编码：
2. 具有散列技巧的字级单热编码：不是明确地为每个单词分配索引并且在字典中保持这些索引的引用，而是可以将单词散列为固定大小的向量。这通常使用非常轻量级的散列函数来完成。

### ****2、using word embeddings****

另一种将矢量与单词关联起来的流行而强大的方法是使用密集的“word vectors”，也称为“word embeddings”。

通过一个热编码获得的向量是二进制的、稀疏的（主要由零组成）和非常高维的（与词汇表中的单词数的维数相同），

单词嵌入是从数据中学习的，是低维浮点向量（即“密集”向量，而不是稀疏向量）。与通过一个热编码获得的字向量不同。在处理非常大的词汇表时，单词嵌入将更多的信息打包到更小的维度中。



有两种方法可以获得单词嵌入：

1. 使用嵌入层学习单词嵌入（Learning word embeddings with the Embedding layer）：

即：与我们关注的主要任务（例如文档分类或情绪预测）一起学习单词嵌入。 在此设置中，将从随机单词向量开始，然后以与学习神经网络权重相同的方式学习单词向量。

将密集向量与单词相关联的最简单方法是随机选择向量。这种方法的问题在于得到的嵌入空间没有结构。深度神经网络很难理解这种非结构化嵌入空间。为了更抽象：单词向量之间的几何关系应该反映这些单词之间的语义关系。 单词嵌入旨在将人类语言映射到几何空间。因此，在每个新任务中学习新的嵌入空间是合理的。反向传播使得这非常容易，而Keras使它变得更加容易。它只是学习图层的权重：嵌入层。

嵌入层至少需要两个参数：标记的数量和嵌入的维度。

嵌入层最好理解为将整数索引（代表特定单词）映射到密集向量的字典。它将这些整数作为输入整数，在内部字典中查找这些整数，并返回相关的向量。这实际上是一个字典查找。

嵌入层接受一个二维的整数张量作为输入，其形状（样本，序列长度），其中每个条目都是一个整数序列。它可以嵌入可变长度的序列，例如，我们可以将具有 shapes (32, 10) (batch of 32 sequences of length 10) or (64, 15) (batch of 64 sequences of length 15)输入嵌入层。但是，一批中的所有序列都必须具有相同的长度（因为我们需要将它们打包成一个张量），所以短于其他序列的序列应该用零填充，长于其他序列的序列应该被截断。

此层返回形状的三维浮点张量（示例、序列长度、嵌入维数）。这样的三维张量可以由RNN层或一维卷积层处理（这两个都将在下一节中介绍）。

当我们实例化一个嵌入层时，它的权重（它的令牌向量的内部字典）最初是随机的，就像任何其他层一样。在训练过程中，这些词向量将通过反向传播逐步调整，将空间构造成下游模型可以利用的内容。一旦完全训练了，你的嵌入空间就会显示出很多结构——一种专门针对你训练模型的特定问题的结构。

让我们将这个想法应用到我们已经熟悉的IMDB电影评论情绪预测任务中。让我们快速准备数据。我们将把电影评论限制在前10000个最常见的词（就像我们第一次使用这个数据集时做的那样），并将评论减少到20个词之后。我们的网络只需学习每10000个单词的8维嵌入，将输入的整数序列（2d整数张量）转换为嵌入序列（3d浮点张量），将张量展平为2d，并在顶部训练单个密集层进行分类。

注意，仅仅展平嵌入的序列并在顶部训练单个Dense层会导致模型分别处理输入序列中的每个单词，而不考虑词间关系和结构句子（例如，它可能会同时对待“这部电影是 狗屎“和”这部电影是狗屎“作为负面”评论“）。 在嵌入序列的顶部添加循环层或1D卷积层会更好，以学习将每个序列作为一个整体考虑在内的特征。 这就是我们将在接下来的几节中关注的内容。

1. 使用预先训练的单词嵌入（Using pre-trained word embeddings）：

即：加载到模型中使用与我们尝试解决的机器学习任务不同的机器学习任务预先计算的单词嵌入。

有时，可用的训练数据非常少，无法单独使用我们的数据来学习适当的任务特定的词汇表嵌入。这时可以从已知高度结构化的预先计算的嵌入空间中加载嵌入向量，并展示有用的属性。在自然语言处理中使用预先训练的单词嵌入背后的基本原理与在图像分类中使用预先训练的网络非常相似：我们没有足够的数据可用于学习我们自己的真正强大的功能，但我们希望我们需要相当通用的功能，即常见的视觉功能或语义功能。在这种情况下，重用在不同问题上学习的特征是有意义的。

有各种预先计算的字嵌入数据库，可以下载并开始在Keras嵌入层中使用。 Word2Vec就是其中之一。另一个流行的被称为“GloVe”。它是一种基于分解词共现统计矩阵的嵌入技术。让我们来看看如何开始在Keras模型中使用GloVe嵌入。当然，相同的方法对于Word2Vec嵌入或您可以下载的任何其他字嵌入数据库都是有效的。我们还将使用此示例来刷新我们在几段前介绍的文本标记化技术：我们将从原始文本开始。把它们放在一起：从原始文本到单词嵌入。我们将使用类似于我们刚刚过去的模型 - 在矢量序列中嵌入句子，展平它们并在顶部训练密集层。 但我们将使用预先训练的字嵌入来实现，而不是使用Keras中打包的预标记化IMDB数据，我们将从头开始，通过下载原始文本数据。步骤如下：

1. 下载IMDB数据作为原始文本

http://ai.stanford.edu/~amaas/data/sentiment/

1. 标记化数据

对我们收集的文本进行矢量化，并准备训练和验证分割。

因为预训练的单词嵌入对于几乎没有可用训练数据的问题特别有用，我们将添加以下扭曲：我们将训练数据限制在其前200样本。 因此，在查看了200个例子后，我们将学习如何对电影评论进行分类......

1. 下载GloVe单词嵌入

https://nlp.stanford.edu/projects/glove/

1. 预处理嵌入
2. 定义模型
3. 将GloVe嵌入加载到模型中
4. 训练和评估

### 3、总结

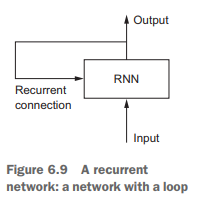
学习了以下内容：

1. 将原始文本转换为神经网络可以处理的内容
2. 使用keras模型中的嵌入层学习特定于任务的令牌嵌入
3. 使用预先培训的单词嵌入，以在小的自然语言处理问题上获得额外的提升。

## 二、了解循环神经网络

密集连接的网络和网络的一个主要特征就是它们没有内存。显示给它们的每个输入都是独立处理的，输入之间没有保持状态。 使用此类网络，为了处理序列或时间序列的数据点，必须立即向网络显示整个序列：将其转换为单个数据点。这就是你在IMDB示例中所做的：整个电影评论被转换为单个大型矢量并一次处理。 这种网络称为前馈网络。相比之下，生物智能以递增的方式处理信息，同时保持其处理内部模型，根据过去的信息建立并随着新信息的不断更新而不断更新。

一个递归神经网络（RNN）采用了同样的原理，尽管是在一个极其简化的版本中：它通过迭代序列元素来处理序列，并保持一个包含与迄今为止所看到的相关信息的状态。实际上，RNN是一种具有内部回路的神经网络（见图6.9）。RNN的状态在处理两个不同的独立序列（如两个不同的IMDB审查）之间重置，因此我们仍然将一个序列视为单个数据点：网络的单个输入。改变的是，这个数据点不再在一个步骤中处理；相反，网络在内部循环序列元素。在许多情况下，不需要这个完整的输出序列; 你只需要最后一个输出（循环结束时的output\_t），因为它已经包含有关整个序列的信息。

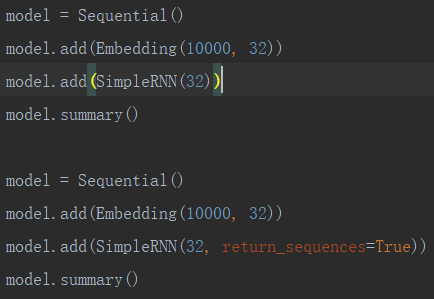


### 1、A recurrent layer in Keras

SimpleRNN 可以以两种不同的模式运行：它可以返回每个时间步的连续输出的完整序列（一个三维形状张量（批量大小、时间步、输出特征）），或者它只能返回每个输入序列的最后一个输出（一个二维张量形状R（批量大小，输出特征）。这两种模式由返回序列构造函数参数控制。在IMDB电影评论分类问题上使用这样一个模型的步骤：

1. Preparing the IMDB data
2. Training the model with Embedding and SimpleRNN layers
3. Plotting results

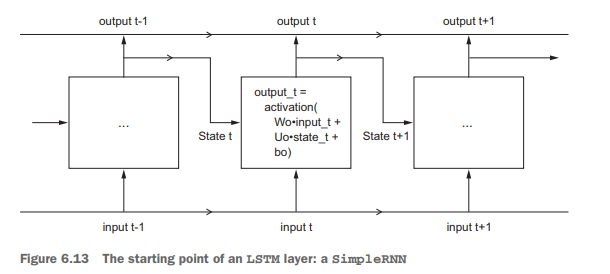
有时，为了增加网络的代表性能力，一个接一个地堆叠几个重复的层是有用的。在这种设置中，必须让所有中间层返回完整序列。在第3章中，我们对这个数据集的第一个方法使我们获得了88%的测试精度。与此基线相比，我们的小循环网络根本没有表现得很好（仅高达85%的验证准确性）。 其中一个问题是我们的输入只考虑前500个字，而不是完整的序列——因此我们的RNN比我们早期的基线模型访问的信息更少。还有就是simpernn不擅长处理长序列，比如文本。其他类型的循环层的性能要好得多。



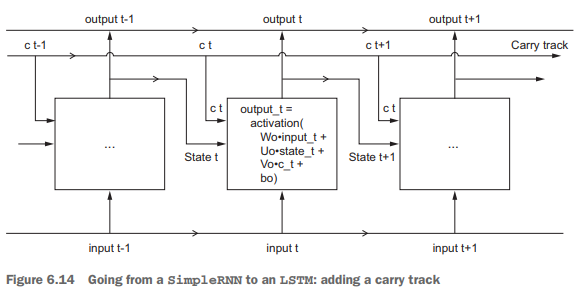
### 2、Understanding the LSTM and GRU layers

SimpleRNN不是Keras中唯一可用的循环层。还有另外两个：LSTM和GRU。在实践中，您将始终使用其中之一，因为SimpleRNN通常过于简单而无法实际使用。 SimpleRNN有一个主要问题：虽然它理论上应该能够在时间t保留关于输入之前看到很多时间步的信息，但实际上，这种长期依赖性是不可能学习的。这是由于消失的梯度问题，这种效应类似于在非常复杂的网络（前馈网络）中观察到的很多层：当不断向网络添加层时，网络最终变得无法控制。LSTM和GRU层旨在解决这个问题。

让我们考虑一下LSTM层。Hochreiter和Schmidhuber于1997年开发了底层的长短期记忆（LSTM）算法；3这是他们在消失梯度问题上的研究成果。这个层是您已经知道的simpernn层的变体；它添加了一种跨多个时间步骤传递信息的方法。想象一条传送带与你处理的顺序平行。从序列中得到的信息可以在任何点跳到传送带上，随后被传送到一个时间步，并在需要时完整地跳下。这基本上就是LSTM所做的：它为以后保存信息，从而防止旧的信号在处理过程中逐渐消失。为了详细了解这一点，让我们从simpernn单元开始（参见图6.13）。因为您将有很多权重矩阵，所以用字母o（wo和uo）索引单元中的w和u矩阵以进行输出。



让我们在这张图中添加一个额外的数据流，它在不同的时间段中传输信息。在不同的时间点调用它的值，其中c代表进位。这一信息将对单元产生以下影响：它将与输入连接和循环连接相结合（通过密集变换：一个加权矩阵的点积，再加上一个偏倚和一个激活函数的应用），它将影响状态发送到下一个时间步（通过激活函数或乘法运算）。从概念上讲，进位数据流是一种调节下一个输出和下一个状态的方法（见图6.14）。



### 3、A concrete LSTM example in Keras

我们将使用LSTM层建立模型并在IMDB数据上进行训练。类似于我们刚刚介绍的SimpleRNN网络。 我们只指定LSTM层的输出维度，并将所有其他参数（有很多）留给Keras默认值。 Keras具有良好的默认值，并且事情几乎总是“正常工作”，而不必花时间手动调整参数。

### 4、总结

本章了解了以下内容：

1）RNN是什么以及它们如何工作

2）什么是LSTM，为什么它在长序列上比简单的RNN更有效

3）如何使用Keras RNN层处理序列数据

接下来，我们将回顾RNN的一些更高级的特性，这些特性可以帮助您充分利用深度学习序列模型。

## 三、递归神经网络的高级应用

在本节中，我们将回顾三种用于改善递归神经网络的性能和泛化能力的先进技术。 到本节结束时，将了解有关使用Keras的循环网络的大部分知识。 我们将展示关于天气预报问题的所有三个概念，我们可以访问安装在建筑物屋顶上的传感器的时间序列数据点，例如温度，气压和湿度，我们用它来预测 收集最后一个数据点后24小时的温度。 这是一个相当具有挑战性的问题，它体现了使用时间序列时遇到的许多常见困难。

我们将介绍以下技术：

1. Recurrent dropout（反复丢失）一种特殊的，内置的方法，使用退出，以防止在重复层过度拟。
2. Stacking recurrent layers（叠加循环层）以增加网络的代表性能力（以较高的计算负载为代价）。
3. Bidirectional recurrent layers（双向循环层）以不同的方式向循环网络提供相同的信息，提高准确性并减轻遗忘问题。

### 1、A temperature forecasting problem

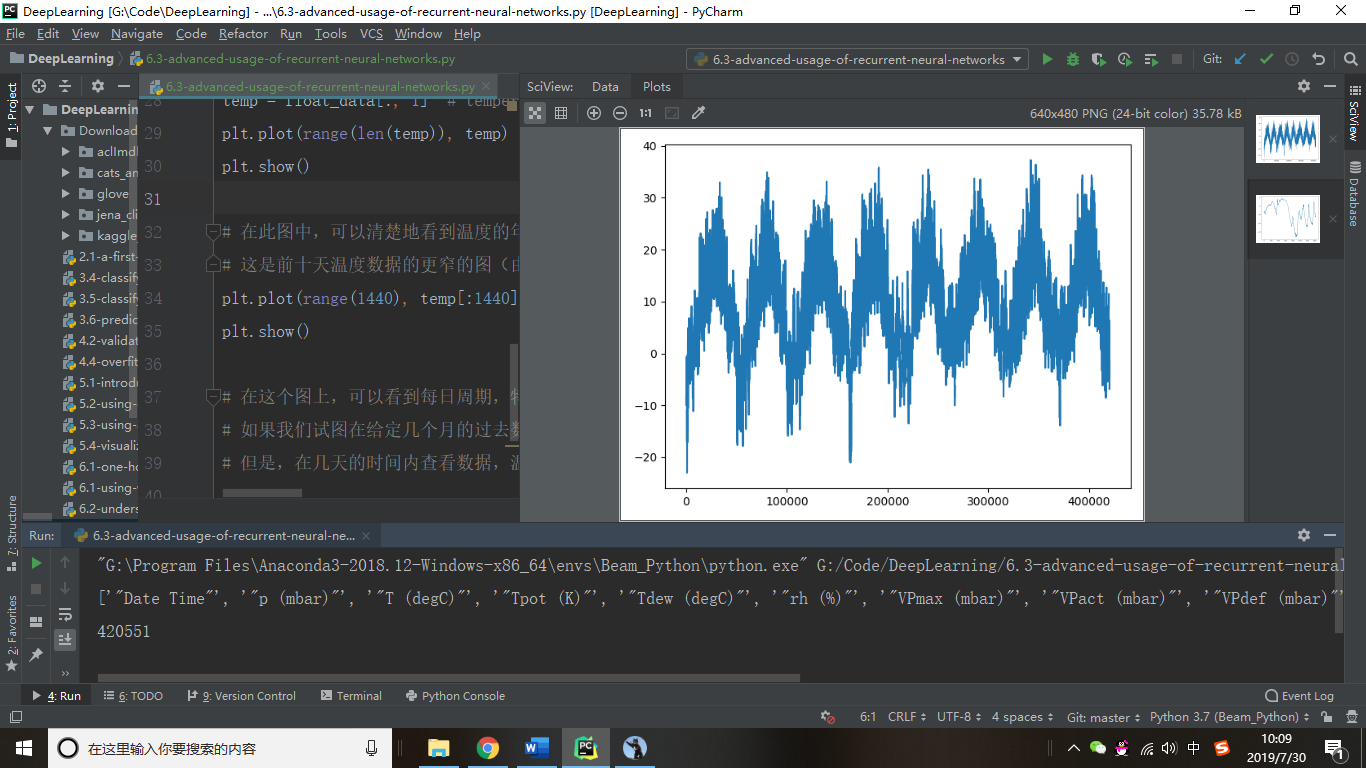
到目前为止，我们所涉及的唯一序列数据是文本数据，例如IMDB数据集和Reuters数据集。但序列数据存在的问题不止语言处理。在本节中的所有示例中，我们将使用在德国Jena的马克斯普朗克生物地球化学研究所气象站记录的天气时间序列数据集：http://www.bgc-jena.mpg.de/wetter/。

在这个数据集中，在过去的几年里，每十分钟记录十四个不同的量（如气温、大气压、湿度、风向等）。此数据集非常适合学习使用数字时间序列。我们将用它建立一个模型，将最近的一些数据（相当于几天的数据点）作为输入，并预测未来24小时的气温。

下载数据<https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip>

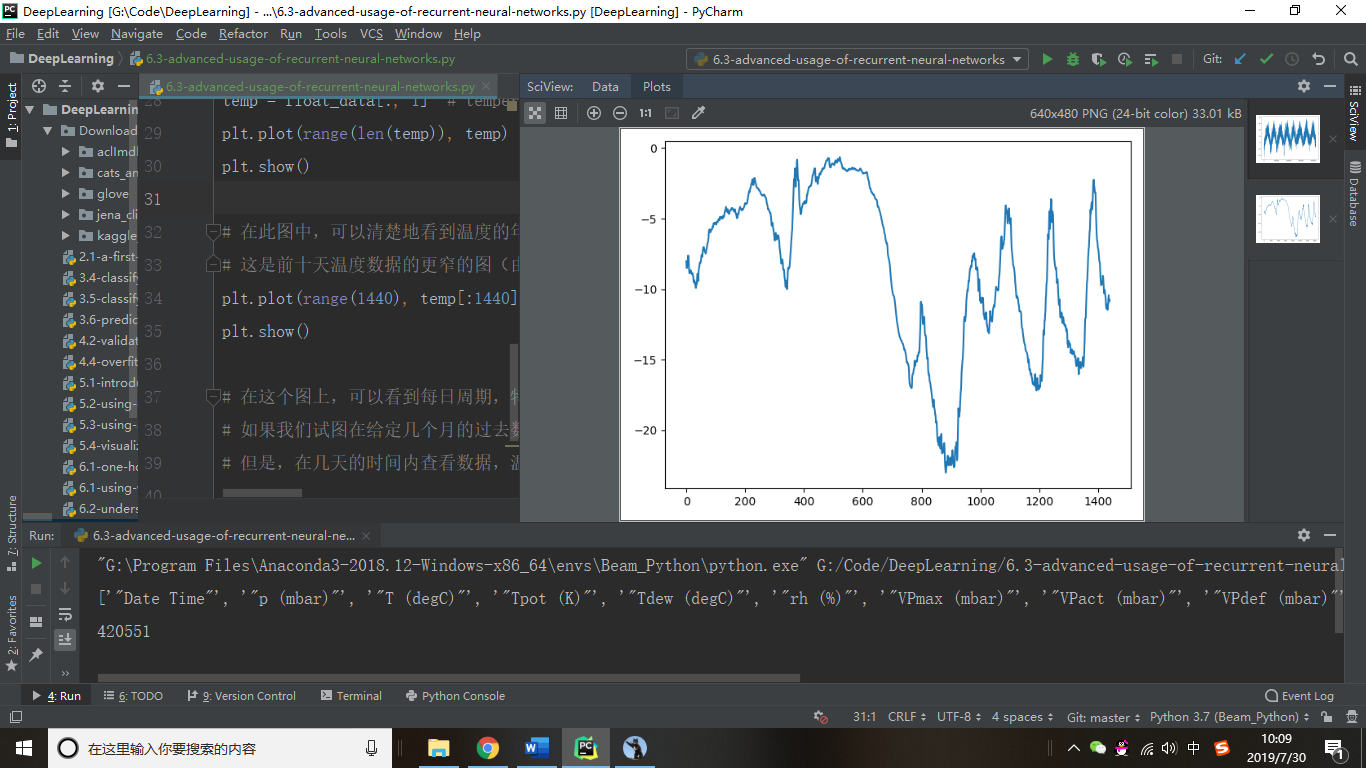
结果：

这是温度（以摄氏度为单位）随时间变化的曲线图：



在此图中，可以清楚地看到温度的年度周期。

这是前十天温度数据的更窄的图（由于数据每十分钟记录一次，我们每天得到144个数据点）：



在这个图上，可以看到每日周期，特别是过去4天。 我们还可以注意到，这十天的时间必须来自一个相当寒冷的冬季。如果我们试图在给定几个月的过去数据的情况下预测下个月的平均温度，由于数据的可靠年度周期性，问题将很容易。但是，在几天的时间内查看数据，温度看起来更加混乱。 那么这个时间序列是否可以在日常范围内预测？ 我们来看看。

### 2、Preparing the data

给定的数据可以追溯到回溯时间步长（时间步长为10分钟）并且每个步骤采样时间步长，我们能否预测延迟时间步长的温度？我们将使用以下参数值：



我们需要做两件事：

1）将数据预处理为神经网络可以摄取的格式。这很简单：数据已经是数字的，所以我们不需要做任何矢量化。然而，数据中的每个时间序列具有不同的规模（例如，温度通常在-20和+30之间，但是以mbar测量的压力大约是1000）。因此，我们将独立地对每个时间序列进行标准化，以便它们都以相似的比例获取小值。

2）编写一个Python生成器，它接收我们当前的浮点数据数组，并从最近的过去产生批量数据，以及未来的目标温度。由于我们的数据集中的样本是高度冗余的（例如样本N和样本N + 1将具有共同的大部分时间步长），因此明确分配每个样本将是非常浪费的。相反，我们将使用原始数据动态生成样本。

我们通过减去每个时间序列的平均值并除以标准偏差来预处理数据。我们计划使用前200,000个时间步作为训练数据，因此我们仅计算这部分数据的平均值和标准差。

现在这里是我们将使用的数据生成器。 它产生一个元组（样本，目标），其中样本是一批输入数据，目标是相应的目标温度阵列。 它需要以下参数：

1. data：浮点数据的原始数组，我们刚刚在上面的代码片段中对其进行了规范化。
2. lookback：我们的输入数据应该返回多少次。
3. delay：未来我们的目标应该是多少次。
4. min\_index和max\_index：数据数组中的索引，用于分隔要绘制的时间步长。 这对于保留一部分数据以进行验证以及另一部分用于测试非常有用。
5. shuffle：是否按时间顺序洗牌或抽取样品。
6. batch\_size：每批的样本数。
7. step：我们对数据进行采样的时间段（以时间步长为单位）。 我们将其设置为6以便每小时绘制一个数据点。

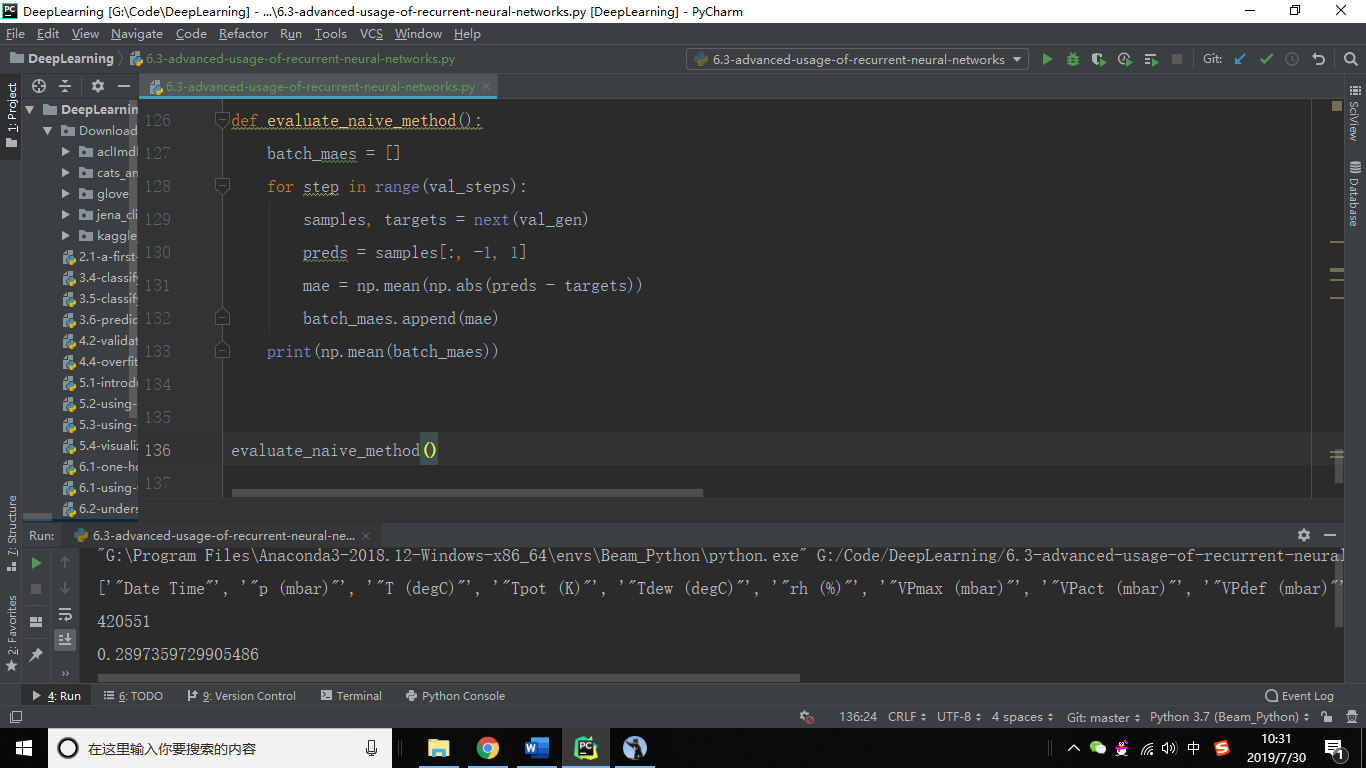
现在让我们使用抽象生成器函数来实例化三个生成器，一个用于训练，一个用于验证，一个用于测试。 每个都将查看原始数据的不同时间段：训练生成器查看前200,000个步骤，验证生成器查看以下100,000个，并且测试生成器查看剩余部分。

### 3、A common-sense, non-machine-learning baseline

在我们开始利用黑盒深度学习模型来解决我们的温度预测问题之前，让我们尝试一种简单的常识方法。它将作为一个完整性检查，它将建立一个我们必须击败的基线，以证明更先进的机器学习模型的有用性。当处理尚未知解决方案的新问题时，这种常识基线非常有用。一个典型的例子是不平衡的分类任务，其中一些类可能比其他类更常见。如果我们的数据集包含90％的A类实例和10％B类实例，那么分类任务的常识方法是在提供新样本时始终预测“A”。这样的分类器总体上将是90％准确，因此任何基于学习的方法都应该超过这个90％的分数以证明有用性。有时这种基本基线可能难以击败。

在我们的例子中，可以假设温度时间序列是连续的（明天的温度可能接近今天的温度）以及每日期间的周期。因此，常识方法是始终预测从现在起24小时的温度将等于现在的温度。让我们使用平均绝对误差度量（MAE）来评估这种方法。平均绝对误差简单地等于：np.mean(np.abs(preds - targets))

写一个评估循环，结果：



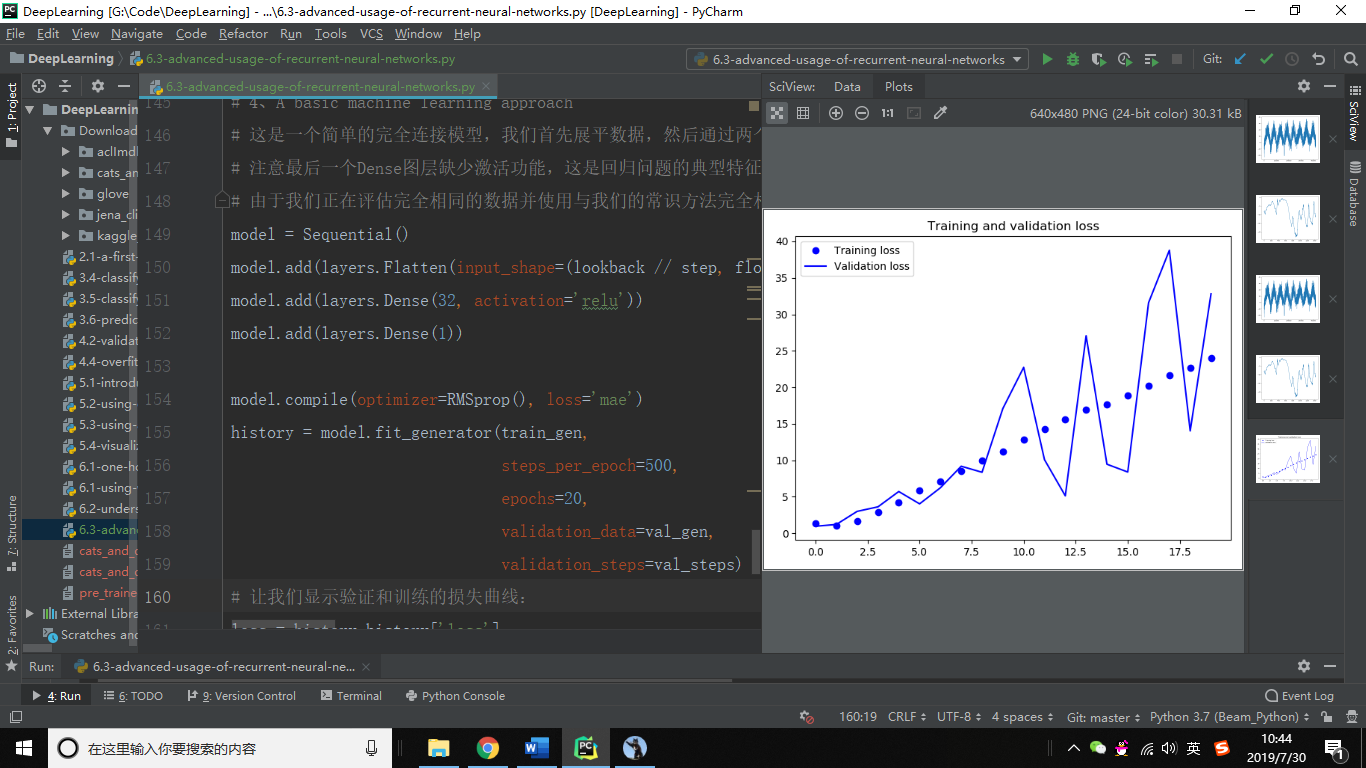
它产生的MAE为0.29。 由于我们的温度数据已经标准化为以0为中心并且标准偏差为1，因此该数字不能立即解释。它转换为平均绝对误差0.29 \* temperature\_std摄氏度，即2.57˚C。 这是一个相当大的平均绝对误差现在游戏是利用我们的深度学习知识做得更好。

### 4、A basic machine learning approach

在尝试机器学习方法之前建立常识基线是有用的，在研究复杂且计算成本高昂的模型（如RNNs） 这是确保我们以后针对该问题提出的任何进一步复杂性是合法的并提供真正好处的最佳方法。

这是一个简单的完全连接模型，我们首先展平数据，然后通过两个密集层运行。请注意最后一个Dense图层缺少激活功能，这是回归问题的典型特征。 我们使用MAE作为损失。 由于我们正在评估完全相同的数据并使用与我们的常识方法完全相同的指标，因此结果将直接具有可比性。

结果，显示验证和训练的损失曲线：



我们的一些验证损失接近无学习基线，但不是非常可靠。这表明首先有这个基线的优点：事实证明并不那么容易。我们的常识包含了机器学习模型无法访问的许多有价值的信息。

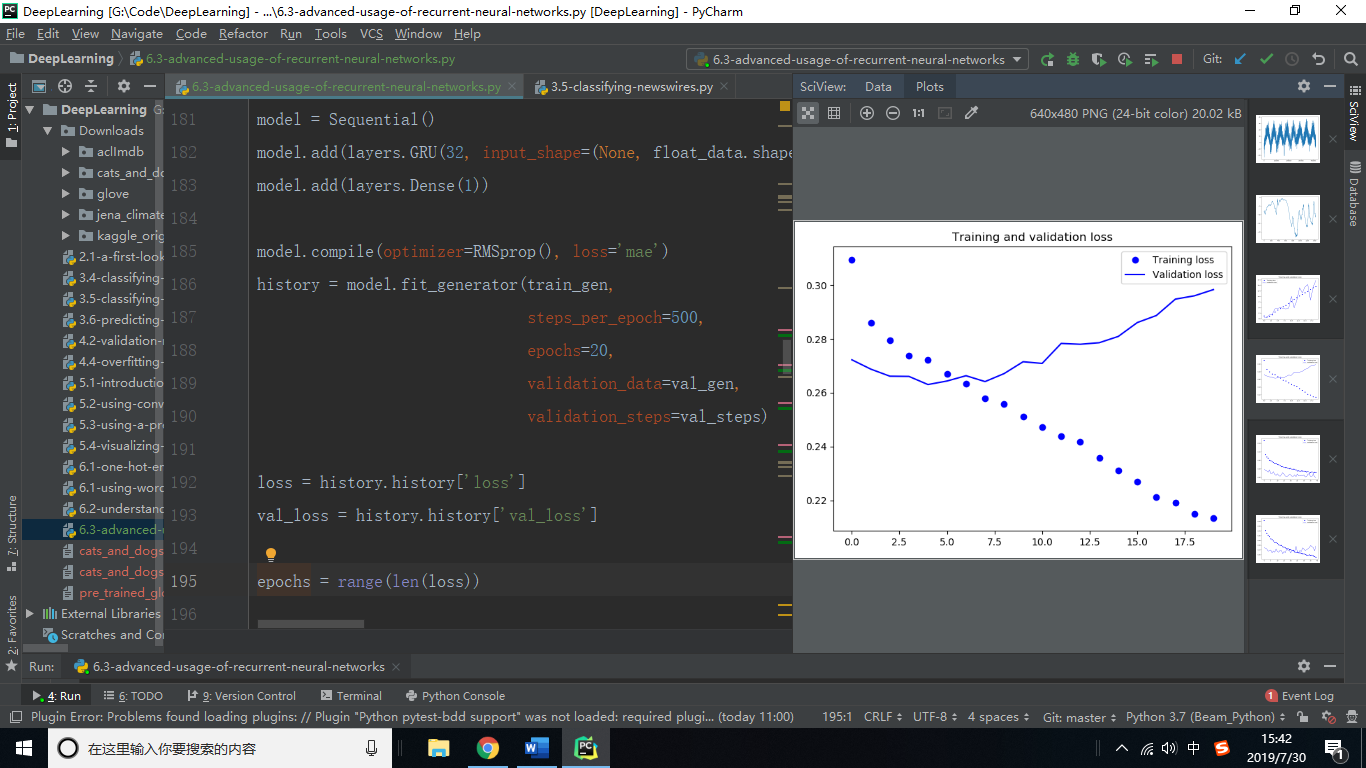
我们可能会有这个疑问：如果存在一个简单，表现良好的模型，从数据到目标（我们的常识基线），为什么我们正在训练的模型不能找到并改进它？简单地说：因为这个简单的解决方案不是我们的训练设置所寻求的。我们在其中寻找解决方案的模型空间，即我们的假设空间，是具有我们定义的配置的所有可能的2层网络的空间。这些网络已经相当复杂。当寻找具有复杂模型空间的解决方案时，简单的良好性能基线可能是无法获得的，即使它在技术上是假设空间的一部分。这是机器学习的一个非常重要的限制：除非学习算法被硬编码以寻找特定类型的简单模型，否则参数学习有时无法找到简单问题的简单解决方案。

### 5、A first recurrent baseline

我们的第一个完全连接的方法并不是那么好，但这并不意味着机器学习不适用于我们的问题。上面的方法包括首先展平时间序列，从输入数据中删除时间概念。让我们来看看我们的数据是什么：一个序列，因果关系和秩序很重要。我们将尝试一种循环序列处理模型 - 它应该是这种序列数据的完美拟合，正是因为它确实利用了数据点的时间排序，这与我们的第一种方法不同。

我们将使用GRU层。 GRU层（代表“门控循环单元”）通过利用与LSTM相同的原理工作，但它们有些简化，因此运行成本更低，尽管它们可能没有LSTM那么多的代表性能力。计算代价和代表性能力之间的这种权衡在机器学习中随处可见。

结果：



好多了！ 我们能够显着击败常识基线，这样就证明了机器学习的价值，以及与此类任务中的序列扁平化密集网络相比，循环网络的优越性。我们新的验证MAE约为0.265（在我们开始显着过度拟合之前）转换为去标准化后的平均绝对误差2.35˚C。 这是我们初始误差2.57˚C的稳固收益，但我们可能还有一些改进余地。

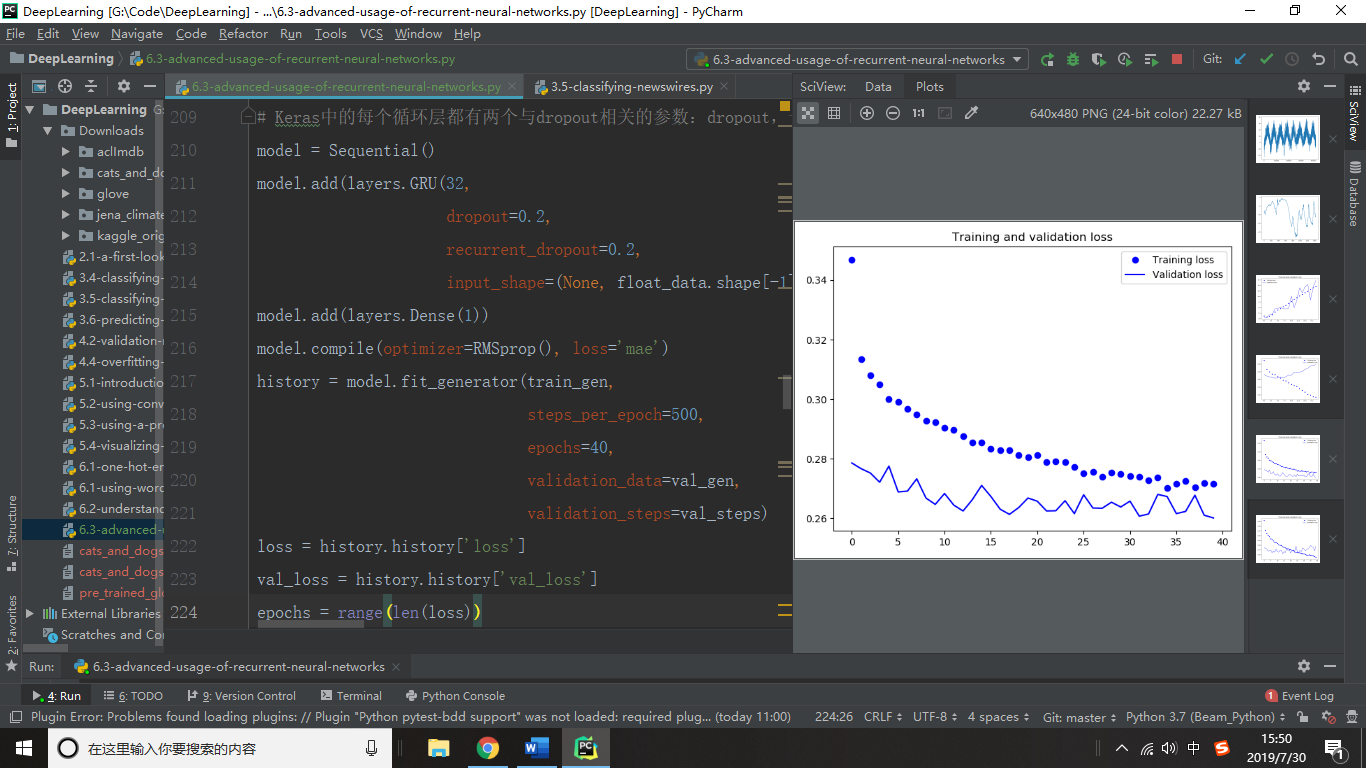
### Using recurrent dropout to fight overfitting

从我们的训练和验证曲线可以看出，我们的模型过度拟合：训练和验证损失在几个时期后开始显着不同。我们已经熟悉了解这种现象的经典技术：丢失，包括随机清零图层的输入单元，以打破图层暴露的训练数据中的偶然相关性。但是，如何在经常性网络中正确应用dropout并不是一个微不足道的问题。长期以来，人们都知道，在反复学习之前申请dropout会阻碍学习，而不是帮助正规化。

使用丢失与循环网络的正确方法：应该在每个时间步应用相同的丢失掩码（相同的丢弃单位模式），而不是从时间步长到时间步长随机变化的丢失掩码。更重要的是：为了规范由GRU和LSTM等层的循环门形成的表示，应将时间上恒定的丢失掩模应用于层的内部循环激活（“循环”丢失掩码）。在每个时间步使用相同的丢失掩码允许网络通过时间正确地传播其学习错误;一个暂时的随机丢失掩码会破坏这个错误信号并对学习过程有害。

Yarin Gal使用Keras进行了他的研究，并帮助将这种机制直接构建到Keras复发层。 Keras中的每个循环层都有两个与dropout相关的参数：dropout，一个指定图层输入单位的丢失率的float，以及recurrent\_dropout，指定循环单位的丢失率。让我们将丢失和重复丢失添加到我们的GRU层，看看它如何影响过度拟合。由于网络正常化与丢失总是需要更长的时间来完全收敛，我们训练我们的网络两倍的时期。

结果：



巨大的成功; 在前30个epods，我们不再过度拟合。然而，虽然我们有更稳定的评估分数，但我们的最佳分数并不比以前低很多。

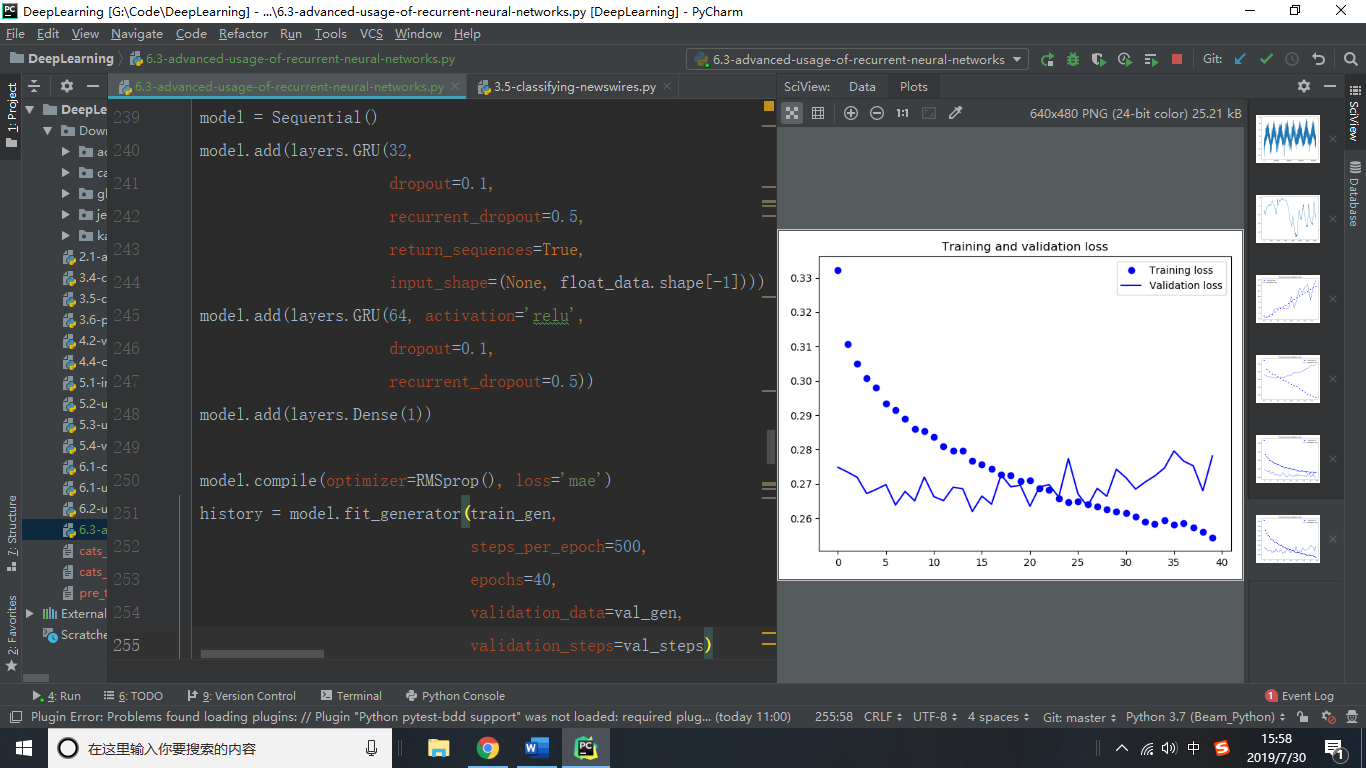
### Stacking recurrent layers

由于我们不再过度拟合，我们似乎遇到了性能瓶颈，我们应该开始考虑增加网络的容量。我们对“通用机器学习工作流程”的描述中学习过：增加网络容量通常是一个好的方法，直​​到过度拟合成为我们的主要障碍（假设您已经采取基本步骤来缓解过度拟合，例如使用退出）。只要你没有过度拟合，那么你的容量可能不足。

通常通过增加层中的单元数或添加更多层来增加网络容量。循环层堆叠是构建更强大的循环网络的经典方法：例如，目前支持谷歌翻译算法的是七个大型LSTM层的堆栈 - 这是巨大的。

要在Keras中将循环层叠加在一起，所有中间层应返回其完整的输出序列（3D张量），而不是在最后一个时间步的输出。这是通过指定return\_sequences = True来完成的。

结果：



我们可以看到，添加的图层确实提高了我们的结果，虽然不是很显着。 我们可以得出两个结论：

1. 由于我们仍然没有过度拟合，我们可以安全地增加层的大小，以寻求一点验证损失改进。 但是，这确实具有不可忽略的计算成本。
2. 由于添加一个层并没有对我们产生重大影响，因此我们可能会看到此时增加网络容量的收益递减。

### Using bidirectional RNNs

双向RNN是常见的RNN变体，其在某些任务上可以提供比常规RNN更高的性能。它经常用于自然语言处理。

RNN特别依赖于顺序或时间依赖性：它们按顺序处理其输入序列的时间步长，并且改组或反转时间步长可以完全改变RNN将从序列中提取的表示。双向RNN利用RNN的顺序灵敏度：它只包含两个常规RNN，例如已熟悉的GRU或LSTM层，每个RNN在一个方向上处理输入序列（按时间顺序和反时间顺序），然​​后合并它们的表示。通过双向处理序列，双向RNN能够捕获可能被单向RNN忽略的模式。

值得注意的是，本节中的RNN层到目前为止按时间顺序处理序列（较早的时间步长）可能是一个随意的决定。如果它是按照反时间顺序处理输入序列，那么我们的RNN可能表现得足够好吗（例如，新的时间步长）？让我们在实践中尝试这一点，看看我们得到了什么。我们需要做的就是编写数据生成器的变体，其中输入序列沿时间维度恢复（最后一行用yield samples[:, ::-1, :], targets替换）。训练与本节第一个实验中使用的相同的一个GRU层网络，我们得到以下结果：

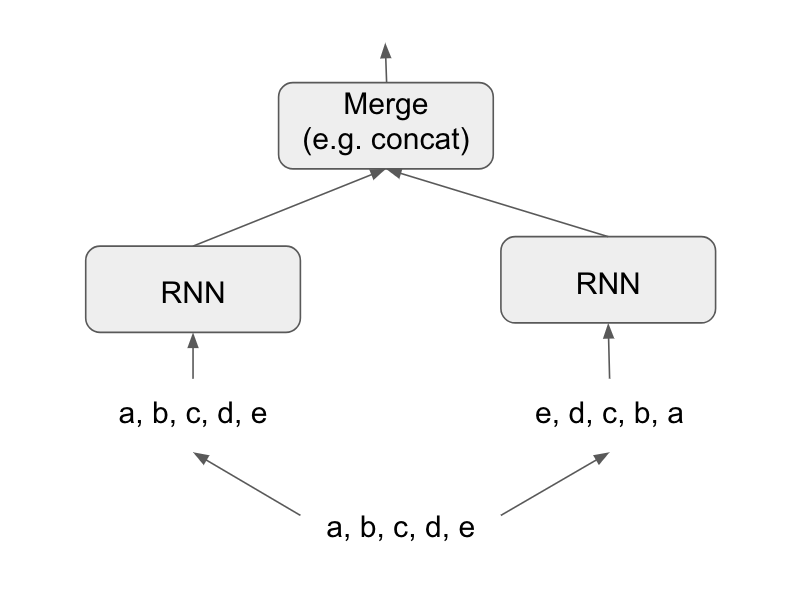


因此，逆序GRU甚至在常识基线上也表现不佳，这表明在我们的案例中按时间顺序处理对于我们方法的成功非常重要。 这是完全合理的：基础GRU层通常会更好地记住最近的过去，而不是遥远的过去，并且自然地，最近的天气数据点比我们的问题中的旧数据点更具预测性（这正是使得常识的原因） 基线相当强的基线）。 因此，层的时间顺序版本必然优于逆序版本。 重要的是，对于许多其他问题，包括自然语言，这通常是不正确的：直观地说，单词在理解句子时的重要性通常不取决于它在句子中的位置。 让我们在上一节的LSTM IMDB示例中尝试相同的技巧。

我们得到的结果与我们在上一节中尝试的时间顺序LSTM几乎相同。

因此，值得注意的是，在这样的文本数据集中，逆序处理与时间顺序处理一样有效，证实了我们的假设，尽管词序在理解语言时很重要，但使用的顺序并不重要。重要的是，对逆向序列进行训练的RNN将比在原始序列上训练的人学习不同的表现形式，就像在现实世界中时间倒流时你会有完全不同的心理模型一样 - 如果你过着生活在你的生活中在你的第一天去世，你出生在你的最后一天。在机器学习中，不同但有用的表示总是值得利用，它们越多越好：它们提供了一个新的角度，可以从中查看数据，捕获其他方法遗漏的数据的各个方面，从而它们可以提高任务的性能。这是“合奏”背后的直觉，我们将在下一章介绍这一概念。

双向RNN利用这一想法来改进按时间顺序的RNN的性能：它以两种方式查看其输入序列，获得可能更丰富的表示并捕获可能仅由时间顺序版本遗漏的模式。



要在Keras中实例化双向RNN，可以使用双向层，其将第一个参数作为循环层实例。 双向将创建此循环层的第二个单独实例，并将使用一个实例按时间顺序处理输入序列，另一个实例以相反的顺序处理输入序列。 让我们试试IMDB情绪分析任务：

它的性能略好于我们在上一节中尝试的常规LSTM，超过88％的验证准确度。 它似乎也更快装配，这并不令人惊讶，因为双向层的参数比按时间顺序的LSTM多两倍。 通过一些正规化，双向方法可能是这项任务的强有力表现者。现在让我们在天气预报任务上尝试相同的方法：

它的表现与普通的GRU层一样。 很容易理解为什么：所有的预测能力必须来自网络的时间顺序一半，因为已知反时间序列的一半在这项任务上严重表现不佳（同样，因为最近的过去比远方的事情重要得多 在这种情况下过去）。

### Going even further

在这个阶段，为了提高我们的天气预报问题的性能，还可以尝试其他许多方法：

1. 调整堆叠设置中每个循环图层中的单位数。我们目前的选择基本上是任意的，因此可能不是最理想的。
2. 调整我们的RMSprop优化器使用的学习速率。
3. 尝试使用LSTM图层而不是GRU图层。
4. 尝试在复发层的顶部使用更大的密集连接的回归量，即更大的Dense层或甚至堆叠的Dense层。
5. 不要忘记最终在测试集上运行性能最佳的模型（在验证MAE方面）！至少您开始开发过度拟合验证集的架构。

深度学习更像是一门艺术，而不是一门科学，虽然我们可以提供关于在某个特定问题上可能起作用或不起作用的指导，但最终每个问题都是独一无二的，你必须尝试根据经验评估不同的策略。 。目前还没有任何理论可以提前告诉您应该采取哪些措施来最佳地解决问题。你必须尝试迭代。

### 10、Wrapping up

本章学习了以下内容:

1）在处理新问题时，最好先为您选择的度量标准建立常识基准。如果没有基线可以击败，无法判断是否取得了任何实际进展。

2）在昂贵的模型之前尝试简单的模型，以证明额外的费用。有时一个简单的模型将成为您的最佳选择。

3）对于时间排序很重要的数据，循环网络非常适合并且容易胜过首先展平时态数据的模型。

4）要将丢失与循环网络一起使用，应使用时间常数丢失掩码和循环丢失掩码。这内置于Keras重复层中，因此您所要做的就是使用循环层的dropout和recurrent\_dropout参数。

5）堆叠的RNN提供比单个RNN层更多的代表性功率。它们也贵得多，因此并不总是值得的。虽然它们在复杂问题（例如机器翻译）上提供了明显的收益，但它们可能并不总是与更小，更简单的问题相关。

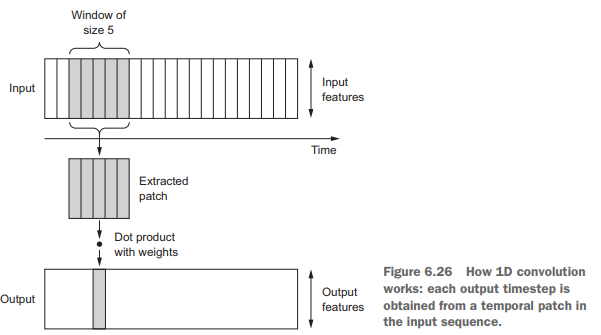
6）双向RNN，以两种方式查看序列，对自然语言处理问题非常有用。然而，它们在序列数据上不是强有力的表现者，其中最近的过去比序列的开始提供更多信息。

请注意，我们在此不会详细介绍两个重要概念：recurrent "attention"和sequence masking。两者都倾向于与自然语言处理特别相关，并不特别适用于我们的温度预测问题。我们将把它们留在本书之外的未来研究中。

## 四、使用convnets进行序列处理

### 1、Understanding 1D convolution for sequence data

先前引入的卷积层是2D卷积，从图像张量中提取2D贴片并对每个贴片应用相同的变换。同样，可以使用1D卷积，从序列中提取局部1D补丁（子序列）（参见图6.26）。



这种一维卷积层可以识别序列中的局部模式。由于在每个补丁上执行相同的输入转换，因此在句子中的某个位置学习的模式稍后可以在不同的位置识别，从而使1d convnets翻译不变（用于暂时翻译）。例如，使用5号卷积窗口的1d convnet字符处理序列应该能够学习长度为5或更小的单词或单词片段，并且应该能够在输入序列的任何上下文中识别这些单词。因此，字符级1d convnet能够学习单词的形态学。

### 2、1D pooling for sequence data

我们已经熟悉2D池操作，例如2D平均池和最大池，用于对网络进行空间下采样图像张量。 2D池操作具有1D等效：从输入提取1D补丁（子序列）并输出最大值（最大池）或平均值（平均池）。与2D convnets一样，这用于减少1D的长度 输入（子采样）。

### 3、Implementing a 1D convnet

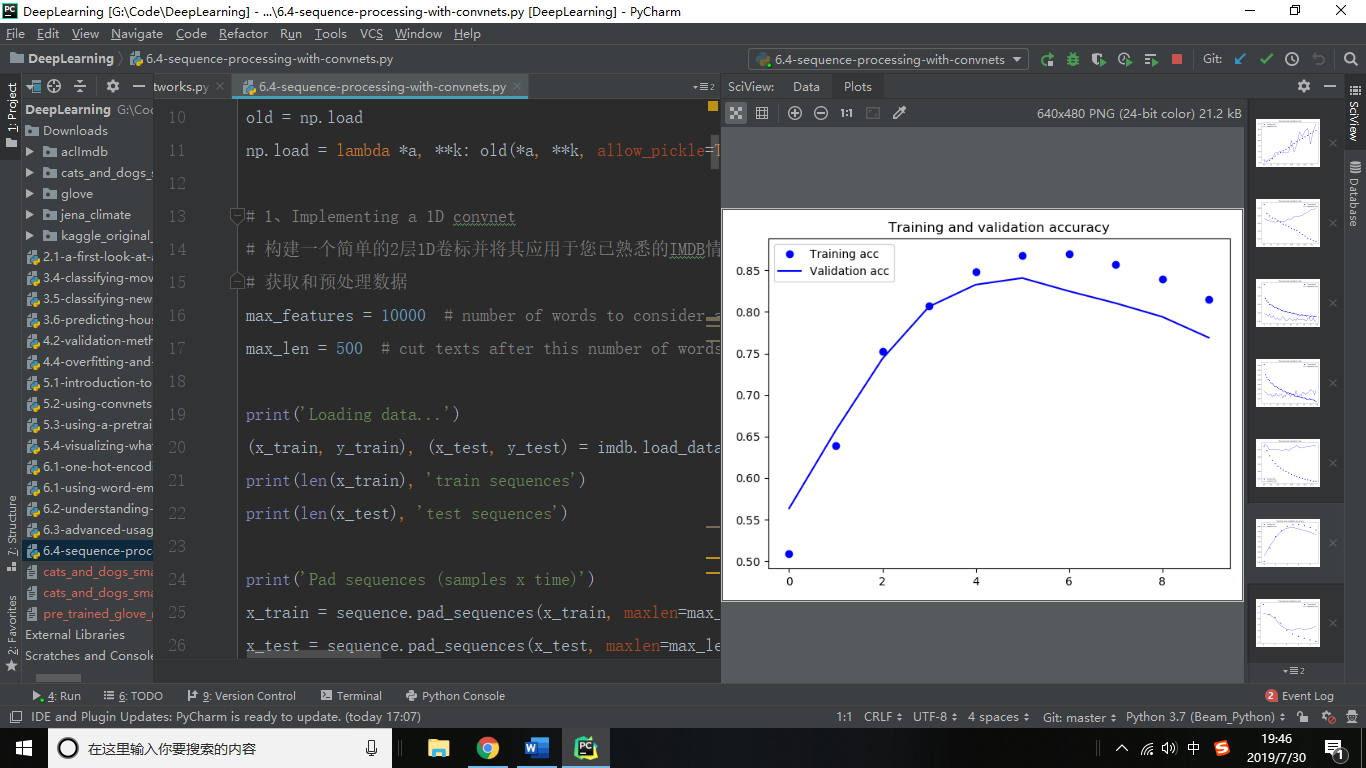
在Keras中，将通过Conv1D层使用1D convnet，它具有与Conv2D非常相似的接口。 它将具有形状（样本，时间，特征）的3D张量作为输入，并且还返回类似形状的3D张量。 卷积窗口是时间轴上的1D窗口，输入张量中的轴1。

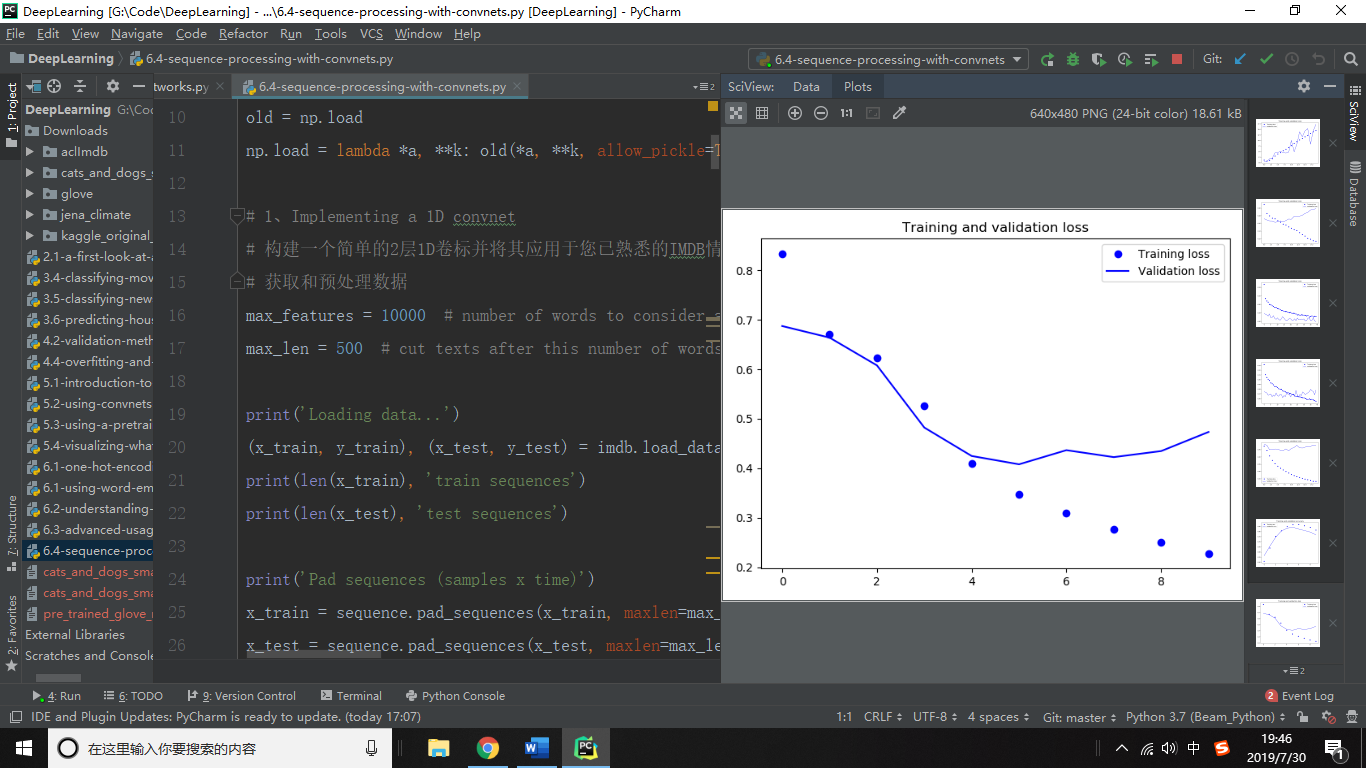
让我们构建一个简单的2层1D convnet 并将其应用于您已熟悉的IMDB情感分类任务。

1D convnets的结构与第5章中使用的2D计数器相同：它们由一堆Conv1D和MaxPooling1D层组成，最终以全局池层或Flatten层结束，转换3D输出 进入2D输出，允许将一个或多个Dense图层添加到模型中，以进行分类或回归。

但是，一个不同之处在于我们可以负担得起使用具有1D小轮的更大卷积窗口的事实。 实际上，对于2D卷积层，3×3卷积窗口包含3 \* 3 = 9个特征向量，但是对于1D卷积层，大小为3的卷积窗口将仅包含3个特征向量。 因此，我们可以轻松地提供尺寸为7或9的1D卷积窗口。

结果：



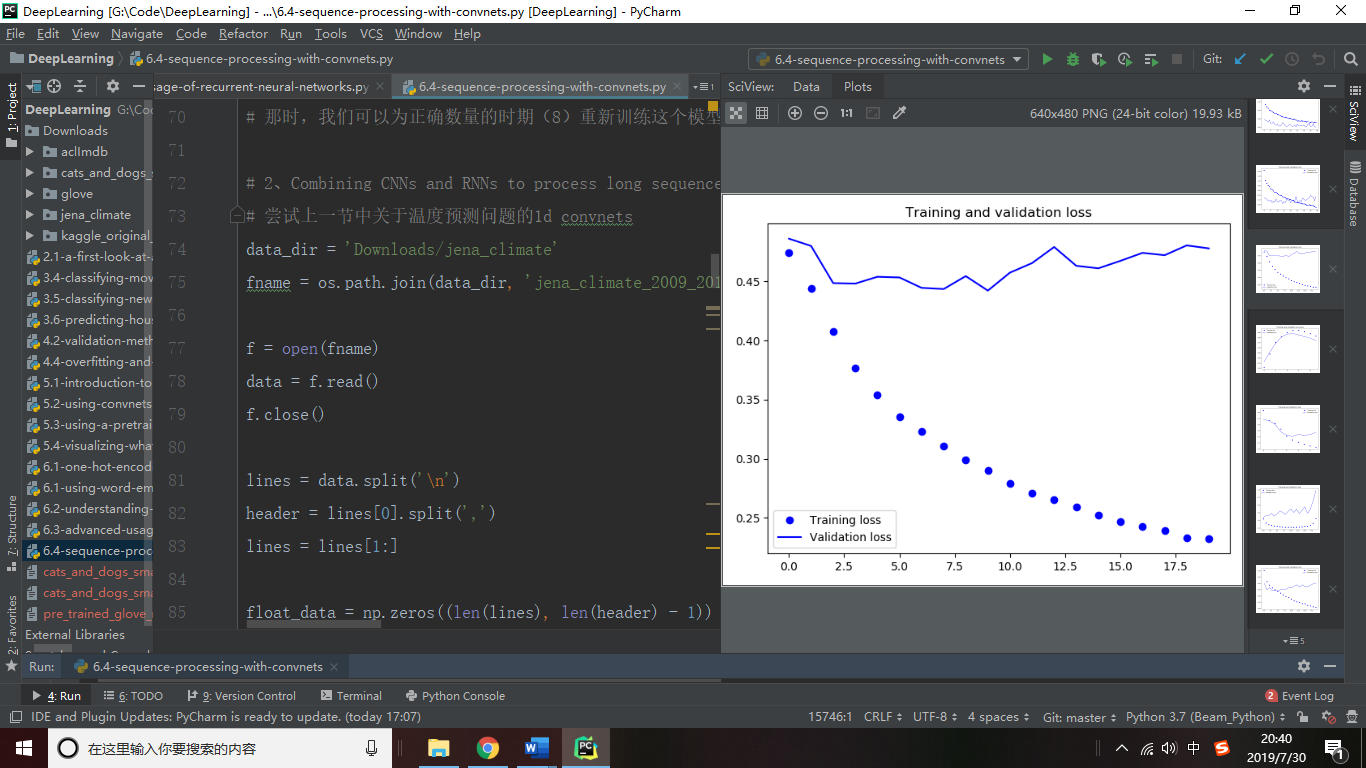


我们的训练和验证结果是：验证准确性略低于我们在两个部分之前使用的LSTM，但运行时间更快，无论是在CPU还是GPU上（尽管确切的加速速度会根据您的确切配置而有很大差异）。 那时，我们可以为正确数量的时期（8）重新训练这个模型，并在测试集上运行它。 这是一个令人信服的证明，一维信号传输可以在字级情绪分类任务上为循环网络提供快速，廉价的替代方案。

### Combining CNNs and RNNs to process long sequences

因为1d convnets独立地处理输入补丁，所以与rnn不同，它们对时间步的顺序（超出局部尺度，卷积窗口的大小）不敏感。当然，为了能够识别长期的模式，我们可以堆叠许多卷积层和聚集层，从而形成上层可以“看到”原始输入的长块——但这仍然是一个相当弱的方法来诱导顺序敏感性。证明这一弱点的一种方法是尝试上一节中关于温度预测问题的1d convnets，其中阶数敏感性是产生良好预测的关键。

结果：



验证MAE保持在0.40s的低位：我们甚至无法使用小型网络打败我们的常识基线。同样，这是因为我们的信号网在输入时间序列中的任何地方查找模式，并且不知道它看到的模式的时间位置（例如，朝向开始，朝向结束等）。由于在此特定预测问题的情况下，更新近的数据点应与旧数据点的解释不同，因此在这里生成有意义的结果时，convnet失败了。这种对网络的限制在IMDB上不是问题，因为与正面或负面情绪相关联的关键字模式将独立于在输入句子中找到它们的位置而提供信息。

将convnet的速度和亮度与RNN的顺序敏感性相结合的一种策略是使用1D convnet作为RNN之前的预处理步骤。当处理那些长时间无法用RNN实际处理的序列时，这尤其有用。数千步的序列。 convnet将长输入序列转换为更短（下采样）的更高级别特征序列。然后，该提取的特征序列成为网络的RNN部分的输入。

这种技术在研究论文和实际应用中并不常见，可能是因为它不是很清楚。它非常有效，应该更常见。让我们试试温度预测数据集。因为这种策略允许我们操纵更长的序列，所以我们可以从后面查看数据（通过增加数据生成器的回溯参数），或者查看高分辨率时间序列（通过减少生成器的步长参数）。在这里，我们将选择（有些任意）使用两倍小的步长，产生两倍的时间序列，其中天气数据以每30分钟一个点的速率被采样。

结果：



从验证损失来看，这种设置不如单独的正规化GRU好，但它的速度要快得多。 它正在查看两次以上的数据，在这种情况下似乎没有太大的帮助，但对其他数据集可能很重要。

### Wrapping up

本章学习了以下内容:

1）与2D convnets在处理2D空间中的视觉模式方面表现良好的方式相同，1D convnets在处理时间模式方面表现良好。 它们在某些问题上提供了更快的RNN替代方案，特别是NLP任务。

2）通常，1D convnets的结构非常类似于计算机视觉领域的2D等价物：它们由Conv1D层和MaxPooling1D层组成，最终以全局池操作或展平操作结束。

3）由于RNN对于处理非常长的序列而言非常昂贵，但是1D convnet是便宜的，因此在RNN之前使用1D convnet作为预处理步骤，缩短序列并提取RNN要处理的有用表示可能是个好主意。

我们在这些页面中将不涉及的一个有用且重要的概念是具有扩张内核的1D卷积。

## 本章总结

1. 在本章中，学习了以下技术，这些技术广泛适用于从文本到时间序列的任何序列数据数据集：
   1. 如何标记文本
   2. 嵌入的是什么词，以及如何使用它们
   3. 什么是recurrent networks，以及如何使用它们
   4. 如何堆叠RNN层并使用双向RNN来构建更强大的序列处理模型
   5. 如何使用1D convnets进行序列处理
   6. 如何组合1D convnets和RNN来处理长序列
2. 您可以使用RNN进行时间序列回归（“预测未来”），时间序列分类，时间序列中的异常检测以及序列标记（例如识别句子中的名称或日期）。
3. 同样，您可以使用1D convnets进行机器翻译（序列 - 序列卷积模型，如SliceNet a），文档分类和拼写校正。
4. 如果全局订单对您的序列数据很重要，那么最好使用循环网络来处理它。这通常是时间序列的情况，其中最近的过去可能比遥远的过去更具信息性。
5. 如果全球排序没有根本意义，那么1D convnets将至少同样有效并且更便宜。对于文本数据通常就是这种情况，其中在句子开头找到的关键字与在结尾处找到的关键字一样有意义。

# Part2-7：Advanced deep-learning best practices

本章包括：

1、Keras功能API

2、使用Keras回调

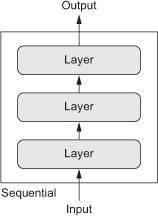
3、使用TensorBoard可视化工具

4、开发最先进模型的重要最佳实践

使用keras函数API，可以构建类似图形的模型，跨不同的输入共享一个层，并像使用python函数一样使用keras模型。Keras回调和基于Tensorboard浏览器的可视化工具允许您在训练期间监控模型。我们还将讨论其他几个最佳实践，包括批标准化、剩余连接、超参数优化和模型集成。

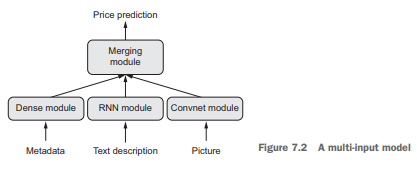
## 一、超越Sequential模型：Keras功能API

到目前为止，本书中介绍的所有神经网络都是使用顺序模型实现的。顺序模型假设网络只有一个输入和一个输出，并且它由一个线性层堆栈组成（见图7.1）。

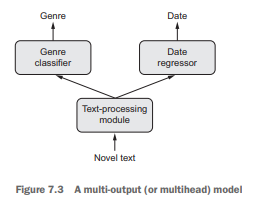


这是一个通常被验证的假设；配置是如此常见，以至于到目前为止，我们只使用顺序模型类就能够涵盖这些页面中的许多主题和实际应用程序。但在许多情况下，这组假设太不灵活了。有些网络需要几个独立的输入，另一些网络需要多个输出，有些网络在层之间有内部分支，使它们看起来像层的图，而不是线性的层堆栈。

例如，有些任务需要多模式输入：它们合并来自不同输入源的数据，使用不同类型的神经层处理每种类型的数据。设想一个深度学习模型，试图使用以下输入来预测二手服装的最有可能的市场价格：用户提供的元数据（如商品的品牌、年龄等）、用户提供的文本描述和商品的图片。如果只有元数据可用，那么可以对其进行热编码，并使用一个紧密连接的网络来预测价格。如果只有文本描述可用，则可以使用RNN或1d convnet。如果你只有这张照片，你可以用一个2d的convnet。但是你怎么能同时使用这三个呢？一个幼稚的方法是训练三个独立的模型，然后对它们的预测进行加权平均。但这可能是次优的，因为模型提取的信息可能是多余的。一个更好的方法是通过使用一个可以同时看到所有可用输入模式的模型来共同学习一个更精确的数据模型：一个具有三个输入分支的模型（见图7.2）。

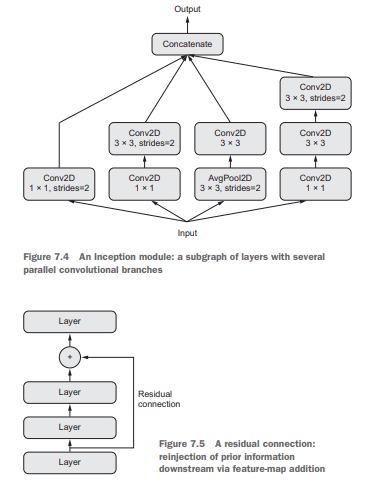


同样，有些任务需要预测输入数据的多个目标属性。考虑到小说或短篇小说的文本，你可能会想自动按流派（如爱情片或惊悚片）对其进行分类，但也会预测它的大概写作日期。当然，你可以训练两个不同的模式：一个为流派，一个为日期。但是，由于这些属性在统计上并不独立，可以通过学习同时联合预测类型和日期来构建一个更好的模型。这样一个联合模型将有两个输出或头（见图7.3）。由于体裁与日期的相关性，了解小说的日期有助于模型学习对小说体裁空间的丰富、准确的表达，反之亦然。



此外，许多最近开发的神经架构需要非线性网络拓扑结构：按有向无环图结构的网络。网络的初始家族（由Szegedy等人开发。例如，在google，依赖于起始模块，其中输入由几个并行卷积分支处理，这些分支的输出随后被合并回一个张量（见图7.4）。

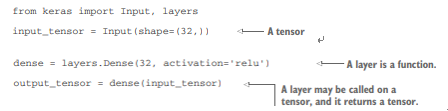
最近还出现了向模型添加剩余连接的趋势，从Resnet网络系列开始（由He等人开发）。在Microsoft中，剩余连接包括通过将过去的输出张量添加到后面的输出张量中，将以前的表示重新注入到下游数据流中（参见图7.5），这有助于防止数据处理流中的信息丢失。还有许多类似图的网络的例子。



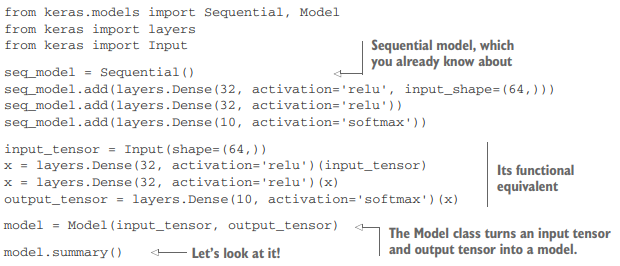
这三个重要的用例——多输入模型、多输出模型和类图模型——在Keras中仅使用顺序模型类是不可能的。但是还有另一种更为通用和灵活的方法来使用keras：函数式API。本节详细解释它是什么，它能做什么，以及如何使用它。

### Introduction to the functional API

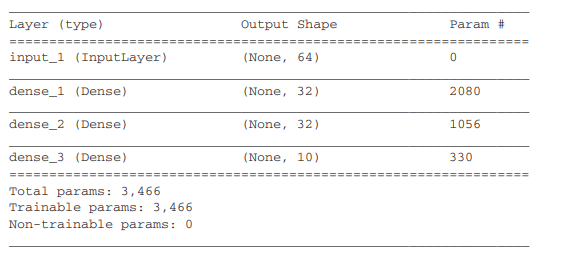
在函数API中，直接操作张量，并使用层作为获取张量和返回张量的函数（因此，称为函数API）：



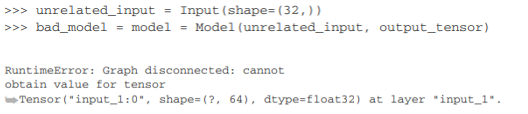
让我们从一个最小的例子开始，这个例子显示了一个简单的顺序模型及其在函数API中的等价物：



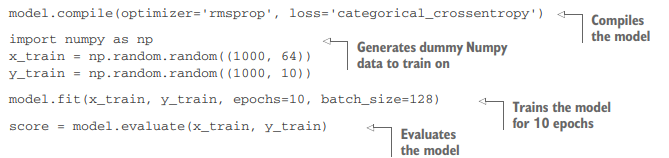
调用model.summary（）将显示以下内容：



此时唯一看起来有点不可思议的部分是仅使用输入张量和输出张量实例化模型对象。在幕后，Keras检索从输入张量到输出张量所涉及的每一个层，将它们组合成一个类似图的数据结构——一个模型。当然，它之所以起作用，是因为输出张量是通过反复变换输入张量得到的。如果试图从与之无关的输入和输出构建模型，则会得到一个运行时错误：



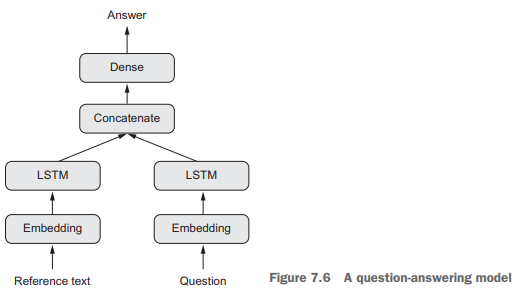
这个错误告诉我们，本质上，Keras无法从提供的输出张量到达input\_1。当涉及到编译、训练或评估此类模型实例时，API与Sequential的API相同：



### Multi-input models

功能API可用于构建具有多个输入的模型。通常，这些模型在某一点上使用一个可以组合多个张量的层合并它们的不同输入分支：通过添加它们、连接它们等等。这通常通过keras合并操作完成，如keras.layers.add、eras.layers.concatenate等。让我们来看一个非常简单的多输入模型示例：问答模型。

典型的问答模型有两个输入：一个自然语言问题和一个文本片段（如新闻文章），提供用于回答问题的信息。然后，该模型必须生成一个答案：在最简单的可能设置中，这是通过SoftMax在某些预定义词汇表上获得的一个单字答案（见图7.6）。



下面是一个如何用函数API构建这样一个模型的示例。设置了两个独立的分支，将文本输入和问题输入编码为表示向量；然后连接这些向量；最后，在连接的represen上添加一个softmax分类器。静态。

双输入问答模型的功能API实现示例~~省略

如何训练两个输入模型？有两种可能的API：可以向模型提供一个numpy数组列表作为输入，或者向它提供一个字典，将输入名称映射到numpy数组。当然，只有在为输入命名时，后一个选项才可用。

### Multi-output models

同样，可以使用功能API来构建具有多个输出（或多个头部）的模型。一个简单的例子是一个网络，它试图同时预测数据的不同属性，例如一个网络，它从一个匿名的人那里输入一系列的社交媒体帖子，并试图预测此人的属性，例如年龄、性别和收入水平（参见图回复7.7）。



训练这种模型需要能够为不同的网络头指定不同的损失函数：例如，年龄预测是一个标量回归任务，而性别预测是一个二元分类任务，需要不同的训练过程。但是，因为梯度下降要求将标量最小化，所以必须将这些损失合并为单个值，以便训练模型。合并不同损失的最简单方法是将它们相加。在Keras中，可以在编译中使用损失列表或损失字典来为不同的输出指定不同的对象；所产生的损失值被汇总为全局损失，在训练期间将其最小化。

注意，非常不平衡的损失贡献将导致模型表示优先针对个人损失最大的任务进行优化，而牺牲了其他任务。为了弥补这一点，可以在对最终损失的贡献中为损失值指定不同的重要性级别。这在损耗值使用不同标度时尤其有用。例如，用于年龄回归任务的均方误差（mse）损失通常取3-5左右的值，而用于性别分类任务的交叉熵损失可低至0.1。在这种情况下，为了平衡不同损失的贡献，可以为交叉熵损失分配10个权重，为MSE损失分配0.25个权重。

### Directed acyclic graphs of layers

使用功能API，不仅可以构建具有多个输入和多个输出的模型，而且还可以实现具有复杂内部拓扑的网络。Keras中的神经网络可以是任意定向的非循环层图。非循环限定符很重要：这些图不能有循环。张量x不可能成为生成x的层之一的输入。唯一允许的处理循环（即循环连接）是循环层内部的。几种常见的神经网络组件被实现为图形。两个值得注意的模块是**初始模块和剩余连接**。为了更好地理解如何使用函数式API来构建层的图表，让我们来看一下如何在keras中实现这两个API。

------

### Layer weight sharing

功能API的一个更重要的特性是能够多次重用层实例。当两次调用一个层实例，而不是为每个调用实例化一个新的层时，将在每次调用中重用相同的权重。这允许我们构建具有共享分支的模型，几个分支共享相同的知识并执行相同的操作。也就是说，它们共享相同的表示，并同时为不同的输入集学习这些表示。

例如，考虑一个试图评估两个句子之间语义相似性的模型。该模型有两个输入（要比较的两个句子），输出的分数介于0和1之间，其中0表示不相关的句子，1表示相同的句子或相互重新表述的句子。这种模型在许多应用程序中都很有用，包括消除对话系统中的自然语言查询。

在这个设置中，两个输入句是可以互换的，因为语义相似度是一个对称的关系：A到B的相似度与B到A的相似度相同。因此，学习两个独立的模型来处理每个输入句是没有意义的。相反，我们希望用一个LSTM层来处理这两个过程。这个LSTM层（其权重）的表示是基于两个输入同时学习的。这就是我们所说的暹罗LSTM模型或共享LSTM。以下是如何在keras函数API中使用层共享（层重用）来实现这种模型：

----

当然，一个层实例可以多次使用，它可以任意调用多次，每次都重用同一组权重。

### Models as layers

重要的是，在功能API中，模型可以像有效地使用层那样被使用，您可以将模型视为“更大的层”。顺序类和模型类都是如此。这意味着您可以对输入张量调用模型并检索输出张量：

y = model(x)

如果模型有多个输入张量和多个输出张量，则应使用张量列表调用：

y1, y2 = model([x1, x2])

当调用模型实例时，你将重用模型的权重，这与调用层实例时所发生的情况完全相同。调用一个实例，无论是层实例还是模型实例，都将始终重用该实例现有的、直观的已学习的表示形式。

通过重用模型实例可以构建的一个简单的实用示例是一个视觉模型，它使用双摄像头作为输入：两个平行摄像头，相距几厘米（一英寸）。这种模型可以感知深度，在许多应用中都很有用。在合并两个提要之前，不需要两个独立的模型来从左摄像头和右摄像头中提取视觉功能。这样的低级处理可以在两个输入之间共享：即，通过使用相同权重的层来完成，从而共享相同的表示。以下是您如何在Keras中实现暹罗视觉模型（共享卷积基础）：---

### Wrapping up

这就结束了我们对Keras函数式API的介绍，它是构建高级深度神经网络体系结构的重要工具。现在我们知道了以下内容：

* 当我们需要任何超过线性层堆栈的东西时，要退出Sequential API
* 如何使用Keras功能API构建具有多个输入、多个输出和复杂内部网络拓扑的Keras模型
* 如何通过多次调用同一层或模型实例，跨不同的处理分支重用层或模型的权重

## 二、使用Keras Callba-Acks和Tensorboard检查和监控深度学习模型

在本节中，我们将回顾如何在训练期间获得对模型内部所发生的事情的更多访问和控制。使用model.fit（）或model.fit生成器（）在一个大数据集上启动一个训练运行，可以有点像启动一架纸飞机：通过初始脉冲，你无法控制它的轨迹或着陆点。如果你想避免不好的结果（从而浪费纸飞机），最好不要使用纸飞机，而是使用能够感知环境、将数据发送回操作员并根据其当前状态自动做出转向决策的无人机。我们在这里介绍的技术将把对model.fit（）的调用从一架纸飞机转换为一架智能、自主的无人机，它可以自我反省并动态地采取行动。

### Using callbacks to act on a model during training

当你训练一个模型的时候，有很多事情你一开始就无法预测，尤其是，你无法判断需要多少时间才能达到最佳的验证损失。到目前为止的例子都采用了训练足够时间的策略，你开始过度适应，用第一次跑步找出合适的时间来训练，然后用这个最佳数字从头开始一次新的训练。当然，这种方法是浪费的。

处理这一问题的更好方法是，当我们的度量验证损失不再改善时，停止训练。这可以通过使用keras回调实现。回调是一个对象（实现特定方法的类实例），它在调用fit时传递给模型，并且在培训期间由模型在各个点调用。它可以访问有关模型状态及其性能的所有可用数据，并可以采取以下操作：中断培训、保存模型、加载不同的权重集或以其他方式更改模型状态。

以下是一些可以使用回调的方法示例：

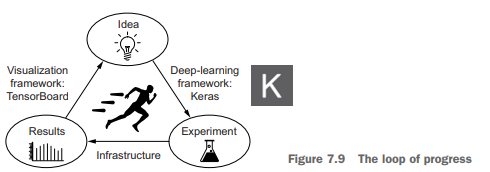
1. 模型检查点 - 在训练期间保存模型在不同点的当前权重。
2. 早期停止 - 当验证损失不再改善时中断培训（当然，保存培训期间获得的最佳模型）。
3. 在训练期间动态调整某些参数的值 - 例如优化器的学习率。
4. 在训练期间记录训练和验证指标，或者可视化模型在更新时学习的表示 – 我们熟悉的Keras进度条是回调！

keras.callbacks模块包含许多内置回调！

### Introduction to TensorBoard: the TensorFlow visualization framework

要做一个好的研究或开发好的模型，你需要对你的模型在实验过程中发生的事情进行丰富的、频繁的反馈。这就是运行实验的要点：获取有关模型如何尽可能多地执行信息的信息。取得进展是一个反复的过程，或者说循环：你从一个想法开始，把它表达为一个实验，试图验证或者使你的想法无效。

你运行这个实验并处理它生成的信息。这激发了你的下一个想法。这个循环的迭代次数越多，你的想法就越精致和强大。Keras可以帮助你在最短的时间内从一个想法过渡到另一个实验，而快速的GPU可以帮助你尽快从一个实验过渡到另一个结果。但是如何处理实验结果呢？TensorBoard就是从这里进来的。



本节介绍TensorBoard，这是一个基于浏览器的可视化工具，与TensorFlow一起打包。 请注意，当您将Keras与TensorFlow后端一起使用时，它仅适用于Keras型号。

  TensorBoard的主要目的是帮助你在训练期间直观地监控模型内部的所有内容。 如果你正在监控的信息不仅仅是模型的最终损失，你可以更清晰地了解模型的功能和不做的事情，并且可以更快地取得进展。 TensorBoard让你可以在浏览器中访问几个简洁的功能：

1. 在培训期间可视化监控指标
2. 可视化您的模型架构
3. 可视化激活和渐变的直方图
4. 探索3D嵌入

通过一个简单的例子来演示这些功能。我们将在IMDB情绪分析任务上训练一维预测。 该模型类似于您在第6章的最后一节中看到的模型。我们将只考虑IMDB词汇表中的前2,000个单词，以使可视化单词嵌入更容易处理。

~~~

### Wrapping up

* Keras回调提供了一种在训练期间监控模型的简单方法，并根据模型的状态自动采取措施。
* 当我们使用TensorFlow时，TensorBoard是一种在浏览器中可视化模型活动的好方法。 可以通过TensorBoard回调在Keras模型中使用它。

## 三、充分利用模型

如果你只是需要一些可以正常工作的东西，盲目地尝试架构就足够好了。在本节中，我们将通过为您提供一套构建最先进的深度学习模型的必备技术的快速指南，从“工作正常”到“工作出色，赢得机器学习比赛”。

### Advanced architecture patterns

在前一节中，我们详细介绍了一个重要的设计模式：残余连接。还有两个您应该了解的设计模式：规范化和非纵向可分离卷积。当你构建高性能的深层convnet时，这些模式尤其相关，但它们也常见于许多其他类型的架构中。

----

### Hyperparameter optimization

在构建深度学习模型时，您必须做出许多看似随意的决策：你应该堆叠多少层？每层应该有多少个单位或过滤器？你应该使用relu作为激活，还是使用不同的功能？您应该在给定图层后使用BatchNormalization吗？你应该使用多少辍学？等等。这些体系结构级参数称为超参数，以将它们与模型的参数区分开来，这些参数通过反向传播进行训练。

 在实践中，经验丰富的机器学习工程师和研究人员随着时间的推移建立直觉，了解在这些选择中哪些有效，哪些无效 - 他们开发了超参数调整技能。但是没有正式的规则。如果你想达到给定任务可以达到的极限，你就不能满足于易犯错误的人做出的任意选择。即使你有很好的直觉，你的初步决定几乎总是不是最理想的。您可以通过手动调整并重复重新训练模型来优化您的选择 - 这就是机器学习工程师和研究人员花费大部分时间做的事情。但是，作为一个人类，你不应该整天摆弄超级参数 - 这最好留给机器。

因此，你需要以原则的方式自动，系统地探索可能的决策空间。 你需要搜索架构空间并根据经验找到最佳的架构空间。 这就是自动超参数优化的领域：它是整个研究领域，也是一个重要领域。 优化超参数的过程通常如下所示：

1）选择一组超参数（自动）。

2）构建相应的模型。

3）使其适合您的训练数据，并测量验证数据的最终性能。

4）选择要尝试的下一组超参数（自动）。

5）重复。

6）最终，测量测试数据的性能。

此过程的关键是在给定各种超参数集的情况下使用此验证性能历史记录的算法，以选择要评估的下一组超参数。许多不同的技术是可能的：贝叶斯优化，遗传算法，简单随机搜索等。

训练模型的权重相对容易：你在一小批数据上计算损失函数，然后使用反向传播算法在正确的方向上移动权重。另一方面，更新超参数极具挑战性。考虑以下：

1）计算反馈信号（这组超参数导致这项任务的高性能模型？）可能非常昂贵：它需要在数据集上从头开始创建和训练新模型。

2）超参数空间通常由离散决策构成，因此不是连续的或可微分的。因此，你通常无法在超参数空间中执行梯度下降。相反，你必须依赖无梯度优化技术，这种技术自然远不如梯度下降。

由于这些挑战很困难，而且这个领域刚兴起不久，我们目前只能使用非常有限的工具来优化模型。通常情况下，随机搜索（选择超参数随机，重复评估）是最好的解决方案，尽管是最天真的解决方案。但是我发现比随机搜索更可靠的一个工具是Hyperopt（https://github.com/hyperopt/hyperopt），这是一个用于超参数优化的Python库，它在内部使用Parzen估计器的树来预测可能有效的超参数集。另一个名为Hyperas的库（https://github.com/maxpumperla/hyperas）集成了Hyperopt，可与Keras模型配合使用。

注意在进行大规模自动超参数优化时要记住的一个重要问题是验证集过度拟合。 因为您基于使用验证数据计算的信号更新超参数，所以你可以有效地对验证数据进行训练，因此它们会快速过度拟合验证数据。 始终牢记这一点。

总的来说，超参数优化是一种强大的技术，绝对需要在任何任务上获得最先进的模型或赢得机器学习竞赛。 想一想：曾几何时，人们手工制作了浅机器学习模型的功能。 这非常不理想。 现在，深度学习使分层特征工程的任务自动化 - 使用反馈信号学习特征，而不是手动调整，这就是应该的方式。 同样，你不应该手工制作你的模型架构; 你应该以原则的方式优化它们。 在撰写本文时，自动超参数优化领域非常年轻和不成熟，就像多年前的深度学习一样，但我预计它将在未来几年内蓬勃发展。

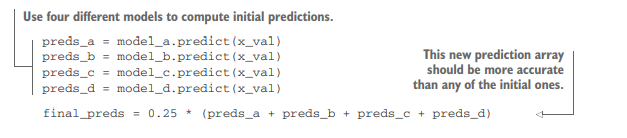
### Model ensembling

另一种在任务中获得最佳结果的强大技术是模型集成。 集成包括将一组不同模型的预测汇集在一起，以产生更好的预测。 如果你看看机器学习比赛，尤其是Kaggle，你会看到获胜者使用非常大的模型集合，无论多么好，都不可避免地击败任何一个模型。

集合依赖于这样的假设，即独立训练的不同优秀模型可能由于不同的原因而变得很好：每个模型都会查看数据的略微不同的方面以进行预测，获得“真相”的一部分但不是全部。你可能熟悉盲人和大象的古代寓言：一群盲人第一次遇到大象并试图通过触摸它来了解大象是什么。每个男人触摸大象身体的不同部分 - 只有一部分，如躯干或腿。然后男人们互相描述大象是什么：“它就像一条蛇，”“就像一根柱子或一棵树”，依此类推。

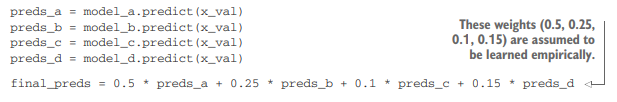
盲人本质上是机器学习模型，试图通过自己的假设（由模型的独特架构和独特的随机权重初始化提供）从各自的角度理解训练数据的多样性。他们每个人都获得了数据真实性的一部分，但并不是全部真相。通过将它们的视角汇集在一起​​，您可以获得更准确的数据描述。大象是各个部分的组合：没有任何一个盲人得到它是正确的，但是，一起采访，他们可以说出一个相当准确的故事。

我们以分类为例。 汇集一组分类器预测（以整合分类器）的最简单方法是在推理时间平均其预测：



只有当分类器或多或少同样好时，这才有效。 如果其中一个明显比其他人差，那么最终的预测可能不如该群体的最佳分类。

集成分类器的一种更智能的方法是进行加权平均，其中在验证数据上学习权重 - 通常，更好的分类器被赋予更高的权重，而更差的分类器被赋予更低的权重。 要搜索一组好的集合权重，你可以使用随机搜索或简单的优化算法，如Nelder-Mead：



有许多可能的变体：例如，您可以对预测的指数进行平均。 通常，在验证数据上优化权重的简单加权平均值提供了非常强的基线。

进行集成工作的关键是分类器的多样性。 多样性是力量。 如果所有的盲人都只触摸了大象的躯干，他们会同意大象像蛇一样，他们会永远不知道大象的真相。 多样化是使集成工作的原因。 在机器学习方面，如果你的所有模型都以相同的方式偏向，那么你的整体将保留同样的偏见。 如果您的模型以不同的方式偏向，则偏差将相互抵消，并且整体将更加稳健和更准确。

因此，你应该在尽可能不同的情况下集合尽可能好的模型。这通常意味着使用非常不同的架构甚至不同品牌的机器学习方法。在很大程度上不值得做的一件事是从不同的随机初始化中独立训练多次相同的网络。如果你的模型之间的唯一区别在于它们的随机初始化以及它们暴露于训练数据的顺序，那么你的整体将是低多样性的，并且仅比任何单个模型提供微小的改进。

我发现在实践中运作良好的一件事 - 但并没有概括到每个问题领域 - 是使用基于树的方法（如随机森林或梯度提升树）和深度神经网络的集合。 2014年，合作伙伴Andrei Kolev和我在Kaggle（www.kaggle.com/c/higgs-boson）的Higgs Boson衰变检测挑战中使用各种树模型和深度神经网络的集合获得第四名。值得注意的是，整体中的一个模型起源于与其他模型不同的方法（它是一个正规化的贪婪森林）并且得分明显低于其他模型。不出所料，它在整体中被分配了一个小重量。但令我们惊讶的是，事实证明它可以通过一个很大的因素来改善整体效果，因为它与其他所有模型都有所不同：它提供了其他模型无法访问的信息。这恰恰是合奏的关键。这不是关于你最好的模型有多好;这是关于你的候选模型集的多样性。

最近，在实践中非常成功的一种基本合奏风格是广泛而深入的模型类别，将深度学习与浅层学习相结合。这些模型包括联合训练深度神经网络和大型线性模型。一系列不同模型的联合培训是实现模型集成的另一种选择。

### Wrapping up

1. 在构建高性能的深层convnet时，需要使用剩余连接、批处理规范化和非纵向可分离卷积。将来，由于具有更高的表示效率，无论是一维、二维还是三维应用，反向可分离卷积都可能完全取代常规卷积。
2. 构建深层网络需要做出许多小的超参数和体系结构选择，这些选择共同决定了模型的好坏。这些选择不是基于直觉或随机机会，而是系统地搜索超参数空间来寻找最优选择。在这个时候，这个过程是昂贵的，而且做它的工具不是很好。但是hyperopt和hyperas库可能会帮助您。在进行超参数优化时，注意验证集过拟合！
3. 赢得机器学习竞赛或以其他方式获得任务的最佳可能结果只能用大量模型来完成。通过一个良好的优化加权平均集成通常是足够好的。记住：多样性就是力量。集合非常相似的模型在很大程度上是毫无意义的；最好的集合是尽可能不同的模型集（当然，尽管有尽可能多的预测能力）。

## 本章总结

在本章中，你学习了以下内容：

* 如何将模型构建为任意的层图形，重用层（层权重共享），并将模型用作Python函数（模型模板化）。
* 你可以使用Keras回调在训练期间监控你的模型，并根据模型状态采取行动。
* TensorBoard允许你可视化度量、激活柱状图，甚至嵌入空间。
* 什么是批标准化、非纵向可分离卷积和剩余连接。
* 为什么要使用超参数优化和模型集成。

有了这些新工具，你可以更好地利用现实世界中的深度学习，并开始构建具有高度竞争力的深度学习模型。

# Part2-8：Generative deep learning

本章包括：

1. 使用LSTM生成文本
2. 实施DeepDream
3. 执行神经风格转移
4. 变分自动编码器
5. 了解生成对抗性网络

在本章中，我们将从多个角度探讨深入学习。我们将回顾序列数据生成（可用于生成文本或音乐）、深度数据流和图像生成，使用变量自动编码器和生成对抗网络。

## 使用LSTM生成文本

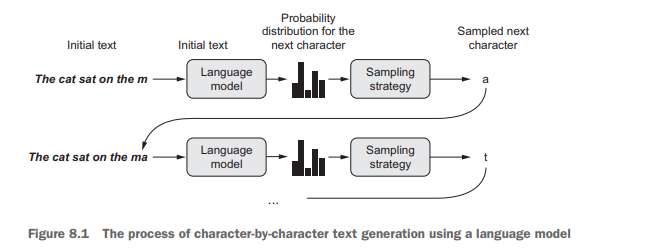
在本节中，我们将探讨如何使用递归神经网络生成序列数据。我们将以文本生成为例，但完全相同的技术可以推广到任何类型的序列数据：可以将它应用到音符序列以生成新的音乐，应用到画笔数据的时间序列（例如，艺术家在iPad上绘画时录制的）到g逐笔生成绘画，等等。序列数据的生成绝不局限于艺术内容的生成。它已成功地应用于语音合成和聊天机器人的对话生成。谷歌在2016年发布的智能回复功能，能够自动生成对电子邮件或文本消息的快速回复选择，由类似的技术提供支持。

### 生成循环网络的简史

### 如何生成序列数据？

在深度学习中生成序列数据的通用方法是训练一个网络（通常是RNN或CONVNET）来预测一个序列中的下一个令牌或下几个令牌，使用前一个令牌作为输入。例如，如果输入“cat在ma上”，网络将被训练来预测目标t，即下一个字符。和通常处理文本数据时一样，标记通常是单词或字符，任何可以为下一个标记建模的网络（给定前一个标记）都称为语言模型。语言模型捕捉语言的潜在空间：它的统计结构。

一旦你有了这样一个经过训练的语言模型，你就可以从中取样（生成新的序列）：你给它一个初始的文本字符串（称为条件数据），让它生成下一个字符或下一个单词（你甚至可以一次生成几个标记），把生成的输出添加回输入数据，并多次重复该过程（见图8.1）。这个循环允许你生成任意长度的序列，这些序列反映了模型所训练的数据的结构：看起来几乎像人类书写的句子的序列。在本节中的示例中，你将获得一个lstm层，向它提供从文本语料库中提取的n个字符的字符串，并训练它预测字符n+1。模型的输出将是所有可能字符的softmax：下一个字符的概率分布。这个LSTM被称为字符级神经语言模型。



### 抽样策略的重要性

生成文本时，选择下一个字符的方式至关重要。一个幼稚的方法是贪婪的抽样，包括总是选择最可能的下一个字符。但这种方法会导致重复的、可预测的字符串，看起来不像连贯的语言。一个更有趣的方法做出了更令人吃惊的选择：它通过从下一个字符的概率分布中抽样，在抽样过程中引入了随机性。这被称为随机抽样（回想一下，随机性就是我们在这个领域所说的随机性）。在这种设置中，如果e有0.3的概率成为下一个字符，根据模型，您将选择30%的时间。请注意，贪婪抽样也可以从概率分布中转换为抽样：一个字符的概率为1，而所有其他字符的概率为0。

从模型的SoftMax输出中进行概率抽样是很好的：它允许在某些时候对甚至不太可能的字符进行抽样，生成更有趣的句子，有时通过提出新的、现实的、在训练中没有出现的发音词来显示创造力。G数据。但是这个策略有一个问题：它没有提供一种方法来控制采样过程中的随机性。

你为什么想要或多或少的随机性？考虑一个极端的例子：纯随机抽样，从一个统一的概率分布中抽取下一个字符，每个字符都是同样可能的。该方案具有最大的随机性，即该概率分布具有最大的熵。当然，它不会产生任何有趣的东西。在另一个极端，贪婪抽样也不会产生任何有趣的结果，也没有随机性：相应的概率分布具有最小的熵。从“实际”概率分布中抽样——模型的SoftMax函数输出的分布构成了这两个极端之间的中间点。但是还有许多其他的中间点，熵的高或低，你可能想探索。熵越小，生成的序列就越具有可预测性（因此它们看起来可能更逼真），而熵越大，生成的序列就越令人惊讶和富有创造性。当从生成模型中取样时，总是很好地探索生成过程中不同数量的随机性。因为我们人类是对生成的数据有多有趣的最终判断者，所以有趣性是非常主观的，而且没有预先知道最佳熵的位置。

为了控制采样过程中的随机性，我们将引入一个名为softmax温度的参数，该参数表征用于采样的概率分布的熵：它表征下一个角色的选择将会出乎意料或可预测的程度。 给定温度值，通过以下列方式重新加权，从原始概率分布（模型的softmax输出）计算新的概率分布。

### 实现字符级LSTM文本生成

让我们将这些想法应用到Keras实现中。你首先需要的是大量的文本数据，你可以用来学习语言模型。你可以使用任何足够大的文本文件或一组文本文件——维基百科、指环王等等。在这个例子中，您将使用尼采的一些著作，尼采是十九世纪末的德国哲学家（翻译成英语）。因此，你将学习的语言模式将是尼采写作风格和选择主题的具体模式，而不是更通用的英语模式。

1. Preparing the data

首先下载语料库并将其转换为小写; 接下来，我们将提取部分重叠的长度maxlen序列，一个热编码，并将它们打包成一个三维numpy数组x形状(sequences, maxlen, unique\_characters)。同时，我们准备了一个包含相应目标的数组Y：在每个提取序列之后出现的一个热编码字符。

1. Building the network

我们的网络是一个单一的LSTM层，然后是一个密集的分类器和所有可能的字符的SoftMax。但是让我们注意到，循环神经网络并不是生成序列数据的唯一方法；1d-convnets在最近的时间里也被证明是非常成功的。由于我们的目标是一个热编码的，我们将使用分类交叉熵作为损失来训练模型。

3) Training the language model and sampling from it

给定一个经过训练的模型和一个种子文本片段，我们通过重复生成新文本：

a）根据目前可用的文本，从模型中绘制下一个字符的概率分布。

b）将分配重新加权到一定的“温度”

c）根据重新加权分布随机抽取下一个字符

d）在可用文本末尾添加新字符

最后，这是我们重复训练和生成文本的循环。我们开始在每一个时代之后使用不同的温度范围生成文本。这使我们能够看到生成的文本是如何随着模型开始聚合而演变的，以及采样策略中温度的影响。

正如我们所看到的，低温会产生极其重复且可预测的文本，但在本地结构非常逼真的情况下：特别是所有单词（一个单词是本地字符模式）都是真正的英语单词。随着温度的升高，生成的文本变得更有趣，令人惊讶，甚至创造性;它有时可能会发明听起来有些合理的新词（例如“eterned”或“troveration”）。在高温下，局部结构开始分解，大多数单词看起来像半随机字符串。毫无疑问，这里的0.5是这个特定设置中文本生成最有趣的温度。始终尝试多种采样策略！学习结构和随机性之间的巧妙平衡是让生成有趣的原因。

请注意，通过训练更大的模型，更长的时间，更多的数据，我们可以获得生成的样本，这些样本看起来比我们的更连贯和更真实。但是，当然，除了随机机会之外，不要期望生成任何有意义的文本：我们所做的只是从统计模型中采样数据，其中字符来自哪些字符。语言是一种通信渠道，通信的内容与通信编码的消息的统计结构之间存在区别。为了证明这种区别，这里有一个思想实验：如果人类语言在压缩通信方面做得更好，就像我们的计算机对我们的大部分数字通信做得更好？那么语言就没那么有意义了，但它缺乏任何内在的统计结构，因此无法像我们一样学习语言模型。

### Wrapping up

1. 我们可以通过训练一个模型来生成离散序列数据，以预测给定先前令牌的下一个令牌。
2. 对于文本，这种模型称为“语言模型”，可以基于单词或字符。
3. 取样下一个标记需要在坚持模型判断的可能性和引入随机性之间取得平衡。
4. 处理这一问题的一种方法是软最高温度的概念。总是在不同的温度下进行实验，以找到“正确”的温度。

## DeepDream

DeepDream是一种艺术图像修改技术，它使用卷积神经网络学习的表示。 它是由谷歌在2015年夏天首次发布的，作为使用Caffe深度学习库编写的实现（这是在TensorFlow首次公开发布之前几个月）。 它很快就变成了互联网的感觉，这要归功于它可能产生的幻觉图片（例如，参见图8.3），充满了算法pareidolia文物，鸟类羽毛和狗眼 - 这是DeepDream网络在ImageNet上训练的副产品 在那里，犬种和鸟类的数量远远超过其中。

DeepDream算法几乎与第5章介绍的convnet过滤器可视化技术完全相同，包括反向运行一个回转：对convnet的输入进行渐变上升，以便最大限度地激活上层的特定过滤器。 新闻报道。 DeepDream使用了同样的想法，只有一些简单的区别：

1. 使用DeepDream，尝试最大化整个层的激活而不是特定过滤器的激活，从而将大量特征的可视化同时混合在一起。
2. 我们不是从空白，略微嘈杂的输入开始，而是从现有图像开始 - 因此产生的效果锁定到预先存在的视觉图案，以某种艺术方式扭曲图像的元素。
3. 输入图像以不同比例（称为八度）进行处理，从而提高可视化的质量。

### 在Keras实施Deepdream

我们将从ImageNet上预先培训的一个预先开始。 在Keras，我们有许多这样的可用网络：VGG16，VGG19，Xception，ResNet50 ......虽然相同的过程可以用于任何这些，但您选择的网络自然会影响您的可视化，因为不同的网络架构会导致不同的学习功能。 原始Deep Dream版本中使用的convnet是一个Inception模型，并且在实践中，已知Inception会产生非常漂亮的Deep Dreams，因此我们将使用Keras附带的InceptionV3模型。

接下来，我们计算“损失”，即在梯度上升过程中我们将寻求最大化的数量。 在第5章中，对于过滤器可视化，我们试图最大化特定层中特定过滤器的值。 在这里，我们将同时最大化多个层中所有过滤器的激活。 具体来说，我们将最大化一组高级层激活的L2范数的加权和。 我们选择的确切图层集（以及它们对最终损失的贡献）对我们将能够生成的视觉效果有很大影响，因此我们希望使这些参数易于配置。 较低层会产生几何图案，而较高层会产生视觉效果，您可以在其中识别ImageNet中的某些类（例如鸟或狗）。 我们将从一个涉及四层的任意配置开始 - 但您肯定希望稍后探索许多不同的配置。

定义一个包含我们损失的张量，即上面列出的层激活的l2范数的加权和。设置梯度上升过程。

Deep Dream算法:

首先，我们定义一个“尺度”列表（也称为“八度音阶”），我们将在其中处理图像。 每个连续的比例比前一个比例大1.4倍（即大40％）：我们首先处理一个小图像，然后我们逐渐升级它。

然后，对于每个连续的比例，从最小到最大，我们运行梯度上升以最大化我们先前在该比例下定义的损失。 在每次渐变上升运行后，我们将得到的图像放大40％。

为了避免在每次连续上标后丢失大量图像细节（导致图像变得越来越模糊或像素化），我们利用一个简单的技巧：每次上标后，我们将丢失的细节重新投射到图像中，这是可能的，因为我们知道原始图像应该是什么样的。规模更大。给定一个小的图像s和一个大的图像大小l，我们可以计算原始图像（假定大于l）调整为大小l和原始调整为大小s之间的差异——这个差异量化了从s到l时丢失的细节。

下面的代码利用了以下简单的辅助numpy函数，所有这些函数都按照它们的名字来做。它们需要安装scipy。

### Wrapping up

1. 样式转换包括创建一个新图像，既保留目标图像的内容，又捕获参考图像的样式。
2. 内容可以通过convnet的高级激活来捕获。
3. 样式可以通过convnet不同层的激活的内部关联来捕获。
4. 因此，深度学习允许使用预先训练的convnet定义的损失将样式转换定义为优化过程。
5. 从这个基本概念开始，许多变体和改进是可能的。

## 神经风格转移

除了DeepDream之外，深度学习驱动图像修改的另一个主要发展是由Leon Gatys等人介绍的神经样式转换。神经式传输算法自最初引入以来经历了许多改进，产生了许多变化，包括一个叫做prisma的病毒智能手机应用程序。为了简单起见，本节重点介绍原始文件中描述的公式。



神经样式转换包括将参考图像的“样式”应用于目标图像，同时保存目标图像的“内容”：

“样式”的含义本质上是图像中的纹理、颜色和视觉模式，在不同的空间尺度上，“内容”是图像的高级宏观结构。例如，在上面的例子中，梵高使用星夜，蓝色和黄色的圆形笔触被认为是“风格”，而图宾根照片中的建筑被认为是“内容”。

风格转换的思想与纹理生成的思想密切相关，在2015年神经风格转换发展之前，在图像处理界有着悠久的历史。然而，事实证明，基于深度学习的风格转换实现提供了前所未有的结果，以前可以实现的经典计算机视觉技术，并引发了计算机视觉创造性应用的惊人复兴。

实现样式转换背后的关键概念与所有深度学习算法的核心思想相同：我们定义一个损失函数来指定我们想要实现的目标，并将这种损失最小化。我们知道我们要实现什么：保留原始图像的“内容”，同时采用参考图像的“样式”。如果我们能够从数学上定义内容和样式，那么一个适当的最小化损失函数将是：



如果距离是一个范数函数（如l2范数），内容是一个获取图像并计算其“内容”表示的函数，样式是一个获取图像并计算其“样式”表示的函数。最小化这种损失将导致样式（生成的图像）接近样式（引用的图像），而内容（生成的图像）接近内容（生成的图像），从而实现我们定义的样式转换。Gatys等人的一个基本观察是，深卷积神经网络提供了精确的数学定义样式和内容函数的方法。让我们看看怎么做。

### 内容丢失

网络中较早层的激活包含有关图像的本地信息，而来自较高层的激活包含越来越多的全局和抽象信息。以不同的方式制定，回旋网的不同层的激活提供了在不同空间尺度上图像内容的分解。因此，我们期望通过回转网的顶层的表示来捕获更全局和更抽象的图像的“内容”。

因此，内容丢失的良好候选者将考虑预先训练的信令，并将在目标图像上计算的顶层激活与在生成的图像上计算的同一层的激活之间的L2范数定义为我们的丢失。 。这将保证，从回旋的顶层看，生成的图像将“看起来与原始目标图像相似”。假设回旋网的顶层看到的确实是输入图像的“内容”，那么这确实可以作为保留图像内容的一种方式。

### 风格损失

虽然内容丢失只利用一个更高层，但风格丢失如Gatys等人所定义。本文利用convnet的多层：我们的目标是在convnet提取的所有空间尺度上捕获样式参考图像的外观，而不仅仅是任何单个尺度。

对于样式丢失，Gatys等人本文利用了一个层激活的“gram矩阵”，即给定层的特征图之间的内积。这个内部积可以理解为表示一个层的特征之间相互关系的映射。这些特征相关性捕捉特定空间尺度模式的统计数据，从经验上来说，这些数据与在该尺度上发现的纹理的外观相对应。

因此，样式丢失的目的是在样式参考图像和生成的图像的不同层的激活中保持类似的内部关联。反过来，这就保证了在不同的空间尺度上找到的纹理在样式参考图像和生成的图像上看起来是相似的。

简而言之，我们可以使用预先训练有素的信号来定义一个损失：

1. 通过在目标内容图像和生成的图像之间保持类似的高级图层激活来保留内容。 小天鹅应该“看到”目标图像和生成的图像“包含相同的东西”。
2. 通过在低级图层和高级图层的激活中保持类似的相关性来保留样式。 实际上，特征相关性捕获纹理：生成的和样式参考图像应该在不同的空间尺度上共享相同的纹理。
3. 现在让我们来看看原始2015神经风格转移算法的Keras实现。 正如您将看到的，它与我们在上一节中开发的Deep Dream实现有很多相似之处。

### Keras的神经风格转移

可以使用任何预先训练的网络实现神经样式转移。 在这里，我们将使用Gatys等人在他们的论文中使用的VGG19网络。 VGG19是我们在第5章介绍的VGG16网络的简单变体，还有三个卷积层。这是我们的一般过程：

1. 设置一个网络，用于同时为样式参考图像，目标图像和生成的图像计算VGG19图层激活。
2. 使用在这三个图像上计算的图层激活来定义上面描述的损失函数，为了实现样式转移，我们将最小化该函数。
3. 设置梯度下降过程以最小化此损失函数。

记住，这项技术所达到的只是一种形式的图像重新纹理，或纹理转移。它最适用于具有强烈纹理和高度自相似的样式参考图像，以及不需要高层次细节以便于识别的内容目标。它通常无法实现相当抽象的专长，例如“将一幅肖像的风格转移到另一幅肖像”。该算法比人工智能更接近于经典信号处理，所以不要期望它像魔术一样工作！

另外，注意，运行这种样式转换算法的速度非常慢。然而，我们的设置操作的转换非常简单，只要有适当的训练数据，就可以通过一个小的、快速的前馈convnet来学习它。因此，通过使用上述方法，首先花费大量的计算周期来生成固定样式参考图像的输入输出训练示例，然后训练一个简单的convnet来学习这种特定样式的转换，就可以实现快速样式转换。一旦完成了这一步，将给定的图像格式化是即时的：它只是这个小convnet的一个向前传递。

### Wrapping up

1. 样式转移包括创建新图像，该图像保留目标图像的“内容”，同时还捕获参考图像的“样式”。
2. “内容”可以通过发布会的高级别激活来捕获。
3. “风格”可以通过预测的不同层的激活的内部相关性来捕获。
4. 因此，深度学习允许将样式转移公式化为使用由预训练的预测定义的损失的优化过程。
5. 从这个基本理念出发，可以进行多种变换和改进！

## 使用变分自动编码器生成图像

## 生成对抗网络简介

## 本章总结

# Part2-9：Conclusions