解析卷积神经网络

-------深度学习实践手册

1. 基础理论篇
2. 实践应用篇

第五章：数据扩充

一般数据扩充方式

水平翻转：horizontally flipping 是数据集扩充一倍

随机扣取（<https://www.kaggle.com/solomonk/pytorch-simplenet-augmentation-cnn-lb-0-945>）：

随机扣取的次数决定了数据集扩充的倍数

增加了卷积神经网络对物体尺度和方向上的鲁棒性

尺度变换：

旋转：

色彩抖动： 颜色空间变化，改变饱和度和明度 plant seed classification比赛中应用颜色空间变化实现对数据集的预处理

**举例，使用keras的ImageData Augmentation图像增广技术：**

Github地址：<https://github.com/leeguandong/DL-data-processing-methods/tree/master/DataAugmentation>

ImageDatagenerator:

keras.preprocessing.image.ImageDataGenerator(

featurewise\_center=False,

samplewise\_center=False,

featurewise\_std\_normalization=False,

samplewise\_std\_normalization=False,

zca\_whitening=False,

zca\_epsilon=1e-6,

rotation\_range=0.,

width\_shift\_range=0.,

height\_shift\_range=0.,

shear\_range=0.,

zoom\_range=0.,

channel\_shift\_range=0.,

fill\_mode='nearest',

cval=0.,

horizontal\_flip=False,

vertical\_flip=False,

rescale=None,

preprocessing\_function=None,

data\_format=K.image\_data\_format())

在每一个batches中产生实时的数据增强。数据会以batch的形式进行循环。

1.featurewise

布尔值，使数据集去中心化，让数据集的均值变成0.

featurewise\_std\_normalization: 让输入的每个样本除以自身标准差。（做高光谱影像预处理时进行了这种处理）

这两个参数都是从数据集整体上对每张图片进行标准化处理。

2.samplewise

布尔值，使输入数据的每个样本均值为0

 samplewise\_std\_normalization: 将输入的每个样本除以自身的标准差。

这个与featurewise不同，前者是从整个数据集的分布去考虑，这个只是针对自身图片，处理效果也有很明显的区别。

1. zca\_whtening

zca白化的作用是针对图片进行PCA降维操作，减少图片的冗余信息，保留最重要的特征。

1. rotation range

指定旋转角度范围，其参数只需要指定一个整数即可，但并不是固定以这个角度进行旋转，而是在[0,指定角度]范围内进行随机角度旋转。

1. width\_shift\_range & height\_shift\_range

width\_shift\_range 是水平方向上的平移，height\_shift\_range 是竖直方向上的平移，其参数可以是[0,1]的浮点数，也可以是大于1，其最大平移距离为图片长或宽的尺寸乘以参数，同样平移距离并不是固定最大平移距离，平移距离在[0,最大平移距离]区间内。

平移图片一般都会超过原图范围，这部分区域会根据fill\_mode的参数来补全。当参数设置过大时，整个原图区域会从给定区域块中消失。

1. shear\_range

错切坐标，效果就是让所有的点的x坐标或者y坐标其中之一保持不变，而对应的y或者x坐标则按比例发生平移，且平移的大小和该点到x轴（或y轴）的垂直距离成正比。

1. zoom\_range

zoom\_range参数可以让图片在长或者宽的方向上进行放大或缩小，可以理解成某方向的resize，因此这个参数可以是一个数或者是一个list。当给出一个数时，图片同时在长宽两个方向进行同等程度的放缩操作。当给出一个list时，则代表[width\_zoom range]

1. channel\_shift\_range

channel\_shift\_range 可以理解成改变图片的颜色，通过对颜色通道的数值的偏移，改变图片整体的颜色。但是这个颜色的改变是在整张图的三个通道上同时实现偏移，像是加了一块有色玻璃在图片前面一样，它不能单独改变某一种颜色，偏移的数值越大，颜色变深的效果就越强。从而把图片一般明显的区域，既要分类的区域地带给凸显出来。

1. horizonal\_flip & vertical\_flip

翻转操作就是在一个维度上进行180度转动。

horizontal\_flip的作用就是随机对图片执行水平翻转操作，意味着并不一定对所有的图片都会执行水平翻转，每次生成均是随机选择图片进行翻转。

Vertical\_flip 竖直翻转。

翻转也并不是适用于所有数据集，动物数据集就不适用，毕竟没有翻转过来的动物。

1. rescale

rescale的作用是对图片的每个像素乘上这个放缩因子，这个操作在所有其他变换操作之前执行，在一些模型当中，直接输入原图的像素值可能会落入激活函数的死亡区，因此设置放缩因子为1/255，把像素值放缩到0-1之间有利于模型的收敛，避免神经元的死亡。

不过图片经过rescale之后，保存到本地的图片用肉眼看是没有任何区别的，但是在内存中直接打印图片的数值，即三基色的一个矩阵，会明显看到每一个维度都在0-1之间。不过需要注意的是如果保存到本地的话，keras依然会将尺寸恢复到0-255之间，只有在内存中打开时才会在0-1之间。

10 fill\_mode

Fill\_mode为填充模式，如前所述，对图片进行平移、放缩错切等操作时，图片中会出现一些缺失的地方。可用该方法进行补全。

参数包括：constant、nearest、reflect、wrap

使用ImageDataGenerator的方法：

fit(x): 根据样本数据进行一些统计信息。

在feature\_center、featurewise\_std\_normalization或者zca\_whitning才用。

fit(x,y): 采用numpy数据和标签数组，在批处理过程中实时产生增强和标准化数据。

flow\_from\_directory(directory): 从目录中导入，在batch中产生实时的增强和标准化数据。

特征的数据扩充方式

Fancy PCA: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

原理：Fancy PCA可以近似的捕获自然图像的一个重要特性，即物体特质与光照强度颜色变化无关。

监督式数据扩充（SDA）：<https://zhuanlan.zhihu.com/p/23249000>

海康威视研究院 2016 场景分类冠军

区别于以物体为中心的图像分类任务，场景分类往往依靠图像整体所蕴含的高层语义进行图像分类。可借助图像标记信息来扩充数据，首先根据原数据训练一个分类的初始模型。而后，利用该模型，对每张图生成对应的特征图和热力图。特征图可指示图像区域与场景标记间的相关概率。之后，可根据此概率映射回原图选择较强相关的图像区域作为扣取的图像块。

题外话：<https://zhuanlan.zhihu.com/p/31012731>

Github地址：<https://github.com/leeguandong/DL-data-processing-methods>

Keras的数据输入方法，配合数据扩充来看，既可以实时扩充，也可以先扩充再用其他框架。

1. 单张图片输入

先从最简单的开始，把图片转成numpy array，单张图片当然只用于预测了。

写模型建议将输入模型模块分离。

1. 多张图片输入

多张图片输入也用于模型预测，把同一个类别的图片文件夹低下所有的图片用for循环load进来，然后加到一个list中，最后concatenate起来。concatenate的作用是把shape为(0, 224, 224, 3)的每张图片tensor，打包成shape为(batch, 224, 224, 3)的tensor，  
这样就能实现批量预测或批量训练了

1. 生成器输入

很多情况下，并不能直接输入数据去训练或者是预测，原因是你的数据集太大了，没办法把所有的图片都载入到内存中去。那keras的data generator就派上用场了，当你的模型需要训练数据的时候，generator会自动从cpu中生成一批图片，送到GPU让模型训练，依次循环，直到训练结束。

使用生成器还可以对生成器的功能进行设计，比如通过生成器可以生成很多数据样本以满足模型输入的需要。

1. 图片尺寸不统一的输入（这种输入方式并不是特别重要的）

问：请问啊，你在 FancyKeras-数据的输入（传统） 这篇文章中提到的输入尺寸不一致的模型，在一个batch上尺寸不一致的模型，可以举个例子吗，或者说是集成模型的话，他们在输入域也有共同的输入尺寸的，在keras源码中是这么体现的，我比较疑惑的是在batch上输入尺寸都不定的模型？？可能没接触到这种，能举个例子帮我理解一下嘛，谢谢呀！！

答：在keras或者是tensorflow，有一个图的概念很重要。我的理解是，模型在训练一个batch的时候，相当于创建了一张新的“计算图”，这张计算图在训练的时候是固定的，无法被修改，当这个batch训练结束后，计算图将重新创建。计算图包含图片输入尺寸的信息，假设确定输入尺寸是224×224×3，那么你的数据集的所有图片都必须先处理成224×224的尺寸，才能执行这个计算图。那么，假如在训练时输入的一个batch（数量大于1）图片里面，有一些图片的尺寸没有处理成224×224，那就无法执行这个计算图了。但如果你的batch size只是1，那么batch与batch之间的图片尺寸可以不一样，因为此时一张计算图只包含一个输入尺寸，而且只需处理一张图片。

这个尺寸指的是模型的输入尺寸不一致，input\_shaoe（None，None，None，3），图像尺寸不一致的话，可以采用归一化操作，输入模型尺寸不一样的话，就不能把不同尺寸的图片打包成一个batch进行输入了，因为这种模型在预测或者训练时只能接受一种尺寸，如果batch里面有多个尺寸，模型就会不知道那个尺寸来计算输出的shape，所以在通常模型计算前要归一化图像尺寸。

目前tensorflow和keras不能变尺寸，在一个batch中做不到尺寸不一致，如果模型要求尺寸不一样只能逐张图片进行输入。预训练模型的输入域都有一个范围，所以在各个预训练的输入尺寸上取交集就行（keras上是这样的）。

* 1. for loop 逐个预测

4.2 利用generator方式进行一张一张输入

Generator还可以设置成多进程多线程的方式来输入数据，能加快训练速度，减少GPU等待时间。

代码：

**Keras：Getting started with the Keras functional API**

函数式编程是定义复杂模型的一种方式，比如多输出模型，直接的非周期性图或者是共享层。

**First example:a densely-connected network**

相比较函数式编程，贯序式太简单了。

1. 每一层的实例都可以以tensor形式被调用，并且返回tensor
2. Input tensor和output tensor可以用来定义model
3. 可以像贯序式model那样进行训练。

Github： <https://github.com/leeguandong/DL-data-processing-methods/blob/master/Keras_Functional_Api/Functional_api.py>

from keras.layers import Input, Dense

from keras.models import Model

# This returns a tensor

inputs = Input(shape=(784,))

*# a layer instance is callable on a tensor, and returns a tensor*

x = Dense(64, activation='relu')(inputs)

x = Dense(64, activation='relu')(x)

predictions = Dense(10, activation='softmax')(x)

*# This creates a model that includes the Input layer and three Dense layers*

model = Model(inputs=inputs, outputs=predictions)

model.compile(optimizer='rmsprop',

loss='categorical\_crossentropy',

metrics=['accuracy'])

model.fit(data, labels) # starts training

**All models are callable,just like layers**

伴随着函数式编程api，很容易去复用训练模型，你可以把任意模型当成层一样的对待，在tensor上进行调用。注意有时候调用模型不仅仅是复用结构，同样也复用weights。

# 解决我之前的疑惑，下面调用model之后就不用写load\_weights操作了，直接就调用了上面的原模型了，

*# This works,and returns the 10-way softmax we defined above*

y = model(x)

去创建一个可以处理序列式输入的模型，你可以把图片分类模型转成成一个视频分类模型，仅需要一行。

关于这层的理解：<http://blog.csdn.net/u012193416/article/details/79477220>

from keras.layers import TimeDistributed

input\_sequences = Input(shape=(20, 784))

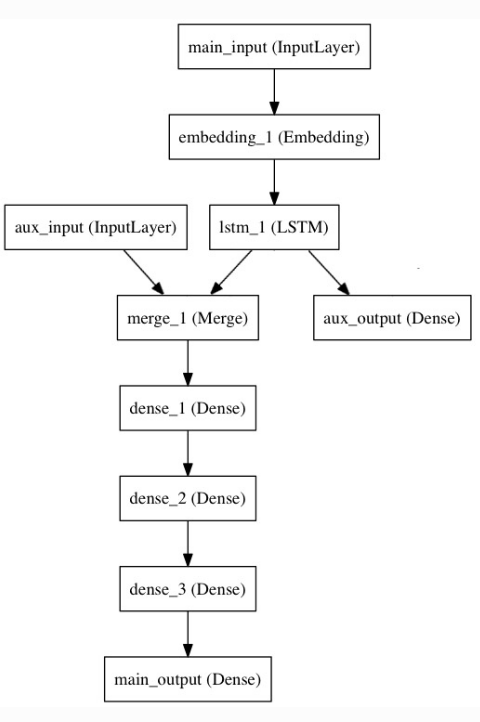
processed\_sequences = TimeDistributed(model)(input\_sequences)

**多输入多输出模型 Multi-input and multi-output models**

函式API使处理大量交织的数据流变得容易。

案例：我们试图预测 Twitter 上的一条新闻标题有多少转发和点赞数。模型的主要输入将是新闻标题本身，即一系列词语，但是为了增添趣味，我们的模型还添加了其他的辅助输入来接收额外的数据，例如新闻标题的发布的时间等。 该模型也将通过两个损失函数进行监督学习。较早地在模型中使用主损失函数，是深度学习模型的一个良好正则方法。

模型结构如下所示：



代码：

<https://github.com/leeguandong/DL-data-processing-methods/blob/master/Keras_Functional_Api/Multi-input%20and%20multi-output%20models.py>

**共享网络层 Shared layers**

使用共享网络层的模型

代码：

<https://github.com/leeguandong/DL-data-processing-methods/blob/master/Keras_Functional_Api/Shared%20layers.py>

**The concept of layer ‘node’**

input\_shape 和 output\_shape 这两个属性也是如此：只要该层只有一个节点，或者只要所有节点具有相同的输入/输出尺寸，那么「层输出/输入尺寸」的概念就被很好地定义，并且将由 layer.output\_shape/layer.input\_shape 返回。但是比如说，如果将一个 Conv2D 层先应用于尺寸为 (32，32，3) 的输入，再应用于尺寸为 (64, 64, 3) 的输入，那么这个层就会有多个输入/输出尺寸，你将不得不通过指定它们所属节点的索引来获取它们。

**More examples**

1. Inception模型
2. 卷积层上的残差连接

from keras.layers import Conv2D, Input

# 输入张量为 3 通道 256x256 图像

x = Input(shape=(256, 256, 3))

# 3 输出通道（与输入通道相同）的 3x3 卷积核

y = Conv2D(3, (3, 3), padding='same')(x)

# 返回 x + y

z = keras.layers.add([x, y])

concatenate和add层的不同：

<http://blog.csdn.net/u012193416/article/details/79479935>

1. 共享视觉模型

两个输入共享一个模型

代码：

<https://github.com/leeguandong/DL-data-processing-methods/blob/master/Keras_Functional_Api/Shared%20vision%20model.py>

1. 视频和图像问答模型

代码：

<https://github.com/leeguandong/DL-data-processing-methods/blob/master/Keras_Functional_Api/Visual%20question%20answering%20model.py>