This script loads the pre-trained Inception v3 model, removes the old top layer, and trains a new one on the flower photos you've downloaded. None of the flower species were in the original ImageNet classes the full network was trained on. The magic of transfer learning is that lower layers that have been trained to distinguish between some objects can be reused for many recognition tasks without any alteration.

这个脚本加载预训练的Inception v3模型，移除旧的顶层，并在下载的鲜花图片上训练一个新的顶层。没有一个花卉种类是在原来的整个网络培训的ImageNet类型。传递学习的神奇之处在于，已经被训练以区分一些对象的较低层可以被重复用于许多识别任务，而不需要任何改变。

The first phase analyzes all the images on disk and calculates the bottleneck values for each of them. 'Bottleneck' is an informal term we often use for the layer just before the final output layer that actually does the classification. This penultimate layer has been trained to output a set of values that's good enough for the classifier to use to distinguish between all the classes it's been asked to recognize. That means it has to be a meaningful and compact summary of the images, since it has to contain enough information for the classifier to make a good choice in a very small set of values. The reason our final layer retraining can work on new classes is that it turns out the kind of information needed to distinguish between all the 1,000 classes in ImageNet is often also useful to distinguish between new kinds of objects.

第一阶段分析磁盘上的所有图片并计算它们中每一个的bottleneck值。Bottleneck是一个非正式术语，我们经常用来描述实际做分类的最后一层的前一层。这个倒数第二层已经被训练输出一组足够好的值，以便分类器用来区分它被要求识别的所有类。这意味着它必须是一个有意义和紧凑的图像总结，因为它必须包含足够的信息，为分类器做出一个很好的选择，在一个非常小的数据集上。我们的最后一层重新训练可以在新类上工作的原因是，事实证明，在ImageNet中所有1000个类别之间区分所需的信息往往也可以用来区分新类型的对象。

Because every image is reused multiple times during training and calculating each bottleneck takes a significant amount of time, it speeds things up to cache these bottleneck values on disk so they don't have to be repeatedly recalculated. By default they're stored in the /tmp/bottleneck directory, and if you rerun the script they'll be reused so you don't have to wait for this part again.   
 由于每个图像在训练过程中都会被多次重复使用，计算每个bottleneck需要花费大量时间，因此将这些bottleneck值缓存在磁盘上可以加快速度，这样就不必重复计算。默认情况下，它们存储在/ tmp / bottleneck目录中，如果你重新运行脚本，它们将被重用，所以你不必再等待这个部分。

Or, if you have a pip installation of tensorflow, `retrain.py` can be run without bazel:

```bash

python tensorflow/examples/image\_retraining/retrain.py \

--image\_dir ~/flower\_photos

```

You can replace the image\_dir argument with any folder containing subfolders of images. The label for each image is taken from the name of the subfolder it's in.

Once the script finishes generating all the bottleneck files, the actual training of the final layer of the network begins.

The training operates efficiently by feeding the cached value for each image into the Bottleneck layer. The true label for each image is also fed into the node labeled GroundTruth. Just these two inputs are enough to calculate the classification probabilities, training updates, and the various performance metrics

一旦脚本完成产生所有的bottleneck文件，网络的最后一层的实际训练就开始了。 通过将每个图像的缓存值提供到Bottleneck层，训练可以高效运行。每个图像的真实标签也被送入标为GroundTruth的节点。只要这两个输入就足以计算分类概率，训练更新和各种性能指标