The model is stored in a python dictionary where each variable name is the key and the corresponding value is a tensor containing that variable's value. To run an image through this network, you just have to feed the image to the model. In TensorFlow, you can do so using the [tf.assign](https://www.tensorflow.org/api_docs/python/tf/assign) function. In particular, you will use the assign function like this:

model["input"].assign(image)

This assigns the image as an input to the model. After this, if you want to access the activations of a particular layer, say layer 4\_2 when the network is run on this image, you would run a TensorFlow session on the correct tensor conv4\_2, as follows:

sess.run(model["conv4\_2"])

模型存储在一个python字典中。只需将一张图片喂给模型就可以通过模型运行该图片。

分派图片作为模型输入。然后，如果想要访问模型特定一层的激活值，例如：运行图片的网络的4\_2层，可以使用：

sess.run(model["conv4\_2"])

We would like the "generated" image G to have similar content as the input image C. Suppose you have chosen some layer's activations to represent the content of an image. In practice, you'll get the most visually pleasing results if you choose a layer in the middle of the network--neither too shallow nor too deep. (After you have finished this exercise, feel free to come back and experiment with using different layers, to see how the results vary.)

我们想要产生和输入图片C相似的图片G。假设选择一些层的激活值来代表一张图片的内容。如果选择网络中既不太浅也不太深的一层，可以得到视觉上最讨人喜欢的结果。（使用不同的层可以得到不同的结果。）

We'll get better results if we "merge" style costs from several different layers

如果我们合并几个不同层的style costs，会得到更好的结果。

STYLE\_LAYERS = [

('conv1\_1', 0.2),

('conv2\_1', 0.2),

('conv3\_1', 0.2),

('conv4\_1', 0.2),

('conv5\_1', 0.2)]

* The style of an image can be represented using the Gram matrix of a hidden layer's activations. However, we get even better results combining this representation from multiple different layers. This is in contrast to the content representation, where usually using just a single hidden layer is sufficient.
* Minimizing the style cost will cause the image GG to follow the style of the image SS.

能够使用一个隐含层的激活值的Gram矩阵来代表图片的风格。但是，结合多个不同层的表现能够得到更好的结果。这与通常只使用一个隐含层就足够的内容画像相反。

最小化style cost会导致图片GG（生成图片）跟随图片SS（风格图片）的风格。

 Unlike a regular session, the "Interactive Session" installs itself as the default session to build a graph. This allows you to run variables without constantly needing to refer to the session object, which simplifies the code.

不像一个常规的session，interactive session以一个默认的构建图的session来安装它自身。这允许我们不需要时时指向一个session对像来运行一个变量。因此，简化了代码。

To get the program to compute the content cost, we will now assign a\_C and a\_G to be the appropriate hidden layer activations. We will use layer conv4\_2 to compute the content cost.

为了让程序计算the concent cost， 我们分派a\_C和a\_G做合适的隐藏层激活值。我们使用conv4\_2层来计算content cost。

a\_G is a tensor and hasn't been evaluated. It will be evaluated and updated at each iteration

a\_G是一个tensor而且还没有被评估。它会在每一次遍历中被评估和更新。

 To get the best looking results, running the optimization algorithm longer (and perhaps with a smaller learning rate) might work better

为了得到最好的视觉效果，运行优化算法更长一些（运行更多的generation）（或许用更小的学习率）