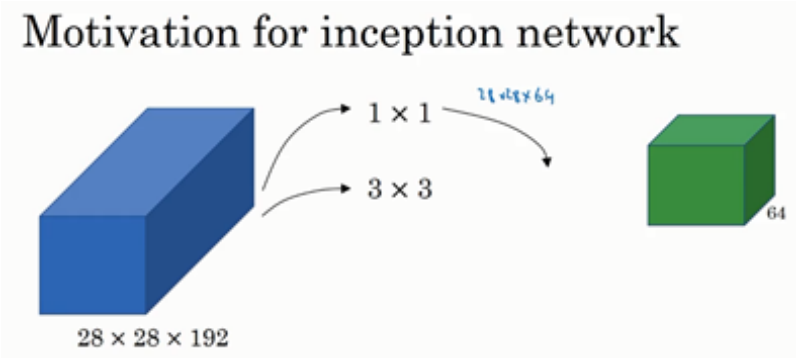
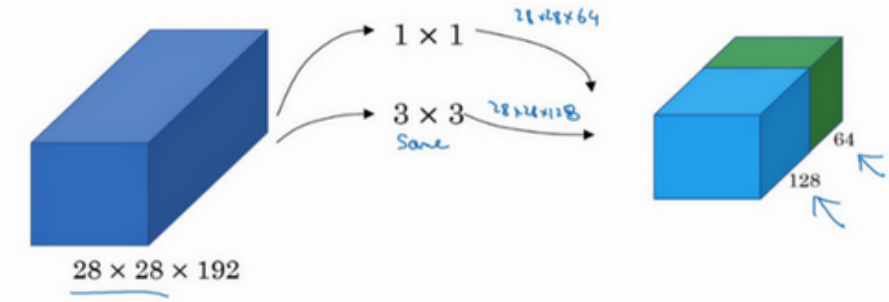
2.6 谷歌 Inception 网络简介（Inception network motivation）

构建卷积层时，你要决定过滤器的大小究竟是 1×1，3×3 还是 5×5，或者要不要添加池化层。**而 Inception 网络的作用就是代替你来决定**，虽然网络架构因此变得更加复杂，但网络表现却非常好，我们来了解一下其中的原理。

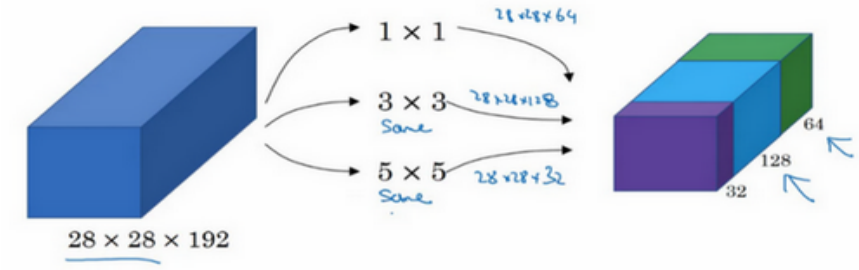
例如，这是你 28×28×192 维度的输入层，**Inception** 网络或 **Inception** 层的作用就是代替人工来确定卷积层中的过滤器类型，或者确定是否需要创建卷积层或池化层，我们演示一下。



如果使用 1×1 卷积，输出结果会是 28×28×#（某个值），假设输出为 28×28×64，并且这里只有一个层。

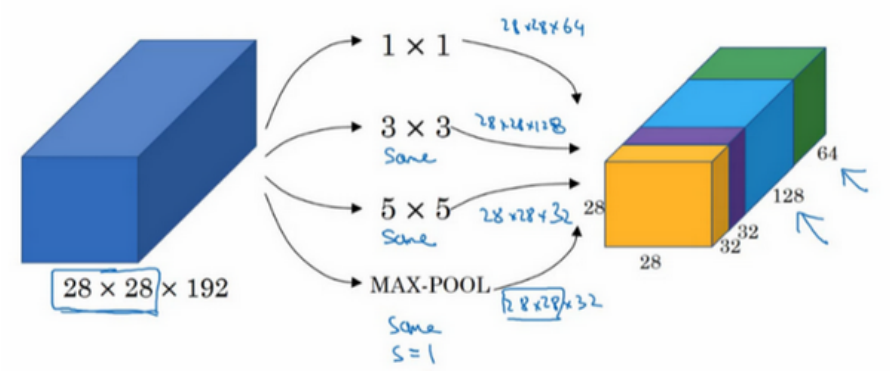


如果使用 3×3 的过滤器，那么输出是 28×28×128。然后我们把第二个值堆积到第一个值上，为了匹配维度，我们应用 **same** 卷积，输出维度依然是 28×28，和输入维度相同，即高度和宽度相同。



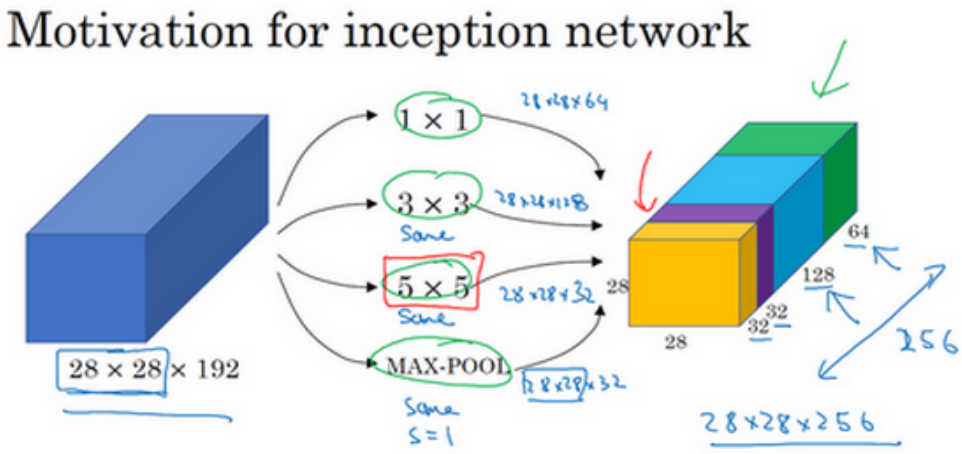
或许你会说，我希望提升网络的表现，用 5×5 过滤器或许会更好，我们不妨试一下，输

出变成 28×28×32，我们再次使用 **same** 卷积，保持维度不变。



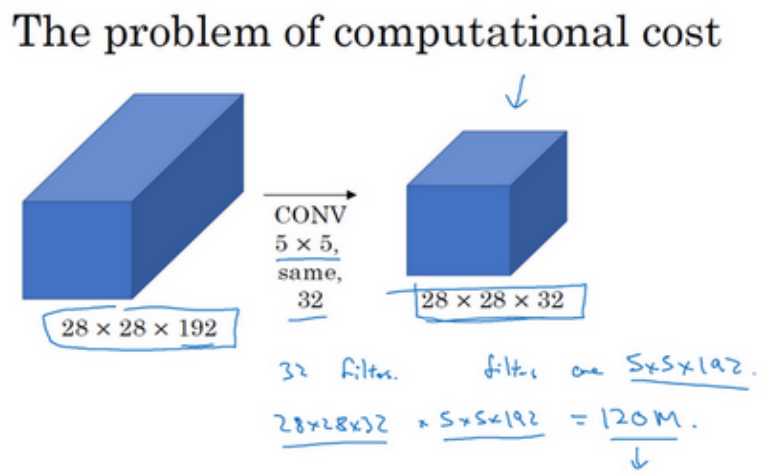
或许你不想要卷积层，那就用池化操作，得到一些不同的输出结果，我们把它也堆积起来，这里的池化输出是 28×28×32。为了匹配所有维度，我们需要对最大池化使用 **padding**，它是一种特殊的池化形式，因为如果输入的高度和宽度为 28×28，则输出的相应维度也是 28×28。然后再进行池化，**padding** 不变，步幅为 1。

这个操作非常有意思，但我们要继续学习后面的内容，一会再实现这个池化过程。

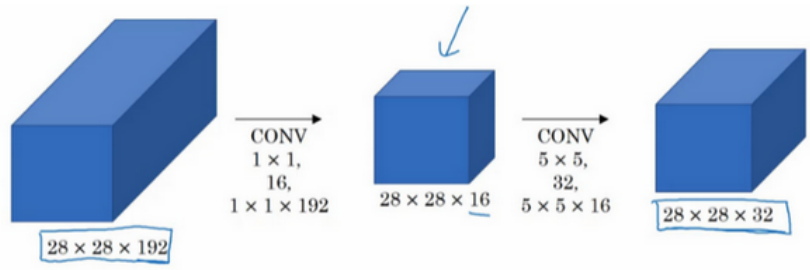


有了这样的 **Inception** 模块，你就可以输入某个量，因为它累加了所有数字，这里的最终输出为 32+32+128+64=256。**Inception** 模块的输入为 28×28×192，输出为 28×28×256。这就是 Inception 网络的核心内容，提出者包括 **Christian Szegedy、刘伟、贾扬清、Pierre Sermanet、Scott Reed、Dragomir Anguelov、Dumitru Erhan、Vincent Vanhoucke** 和 **Andrew Rabinovich**。**基本思想是 Inception 网络不需要人为决定使用哪个过滤器或者是否需要池化，而是由网络自行确定这些参数，你可以给网络添加这些参数的所有可能值，然后把这些输出连接起来，让网络自己学习它需要什么样的参数，采用哪些过滤器组合。**

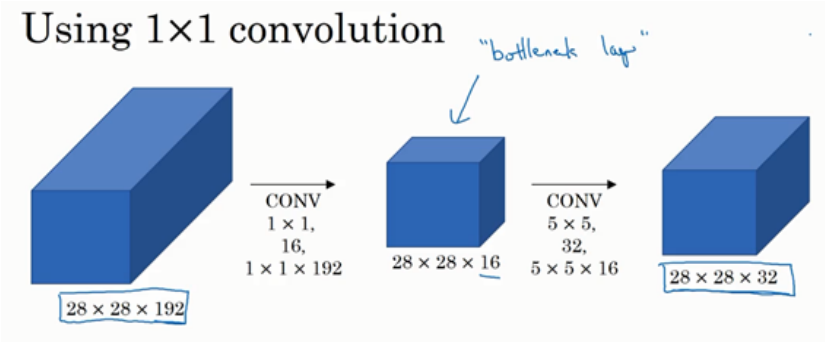
不难发现，我所描述的 **Inception** 层有一个问题，就是计算成本，下一张幻灯片，我们就来计算这个 5×5 过滤器在该模块中的计算成本。



我们把重点集中在前一张幻灯片中的 5×5 的过滤器，这是一个 28×28×192 的输入块，执行一个 5×5 卷积，它有 32 个过滤器，输出为 28×28×32。前一张幻灯片中，我用一个紫色的细长块表示，这里我用一个看起来更普通的蓝色块表示。我们来计算这个 28×28×32 输出的计算成本，它有 32 个过滤器，因为输出有 32 个通道，每个过滤器大小为 5×5×192，输出大小为 28×28×32，所以你要计算 28×28×32 个数字。对于输出中的每个数字来说，你都需要执行 5×5×192 次乘法运算，所以乘法运算的总次数为每个输出值所需要执行的乘法运算次数（5×5×192）乘以输出值个数（28×28×32），把这些数相乘结果等于 1.2 亿（120422400）。即使在现在，用计算机执行 1.2 亿次乘法运算，成本也是相当高的。下一张幻灯片会介绍 1×1 卷积的应用，也就是我们上节课所学的。为了降低计算成本，我们用计算成本除以因子 10，结果它从 1.2 亿减小到原来的十分之一。请记住 120 这个数字，一会还要和下一页看到的数字做对比。



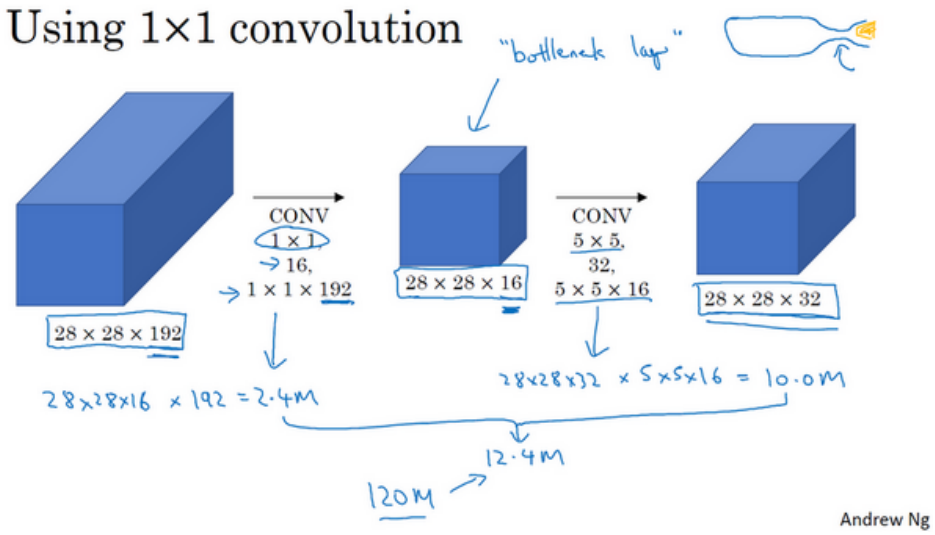
这里还有另外一种架构，其输入为 28×28×192，输出为 28×28×32。其结果是这样的，对于输入层，使用 1×1 卷积把输入值从 192 个通道减少到 16 个通道。然后对这个较小层运行 5×5 卷积，得到最终输出。请注意，输入和输出的维度依然相同，输入是 28×28×192，输出是 28×28×32，和上一页的相同。但我们要做的就是把左边这个大的输入层压缩成这个较小的的中间层，它只有 16 个通道，而不是 192 个。



有时候这被称为瓶颈层，瓶颈通常是某个对象最小的部分，假如你有这样一个玻璃瓶，这是瓶塞位置，瓶颈就是这个瓶子最小的部分。

同理，瓶颈层也是网络中最小的部分，我们先缩小网络表示，然后再扩大它。

接下来我们看看这个计算成本，应用 1×1 卷积，过滤器个数为 16，每个过滤器大小为 1×1×192，这两个维度相匹配（输入通道数与过滤器通道数），28×28×16 这个层的计算成本是，输出 28×28×192 中每个元素都做 192 次乘法，用 1×1×192 来表示，相乘结果约等于 240 万。



那第二个卷积层呢？240 万只是第一个卷积层的计算成本，第二个卷积层的计算成本又是多少呢？这是它的输出，28×28×32，对每个输出值应用一个 5×5×16 维度的过滤器，计算结果为 1000 万。

所以所需要乘法运算的总次数是这两层的计算成本之和，也就是 1204 万，与上一张幻灯片中的值做比较，计算成本从 1.2 亿下降到了原来的十分之一，即 1204 万。所需要的加

法运算与乘法运算的次数近似相等，所以我只统计了乘法运算的次数。

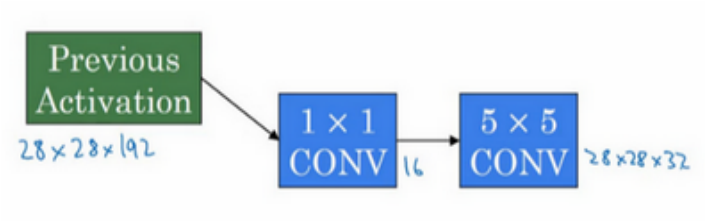
总结一下，如果你在构建神经网络层的时候，不想决定池化层是使用 1×1，3×3 还是 5×5 的过滤器，那么 **Inception** 模块就是最好的选择。我们可以应用各种类型的过滤器，只需要把输出连接起来。之后我们讲到计算成本问题，我们学习了如何通过使用 1×1 卷积来构建瓶颈层，从而大大降低计算成本。

你可能会问，仅仅大幅缩小表示层规模会不会影响神经网络的性能？事实证明，只要合理构建瓶颈层，你既可以显著缩小表示层规模，又不会降低网络性能，从而节省了计算。

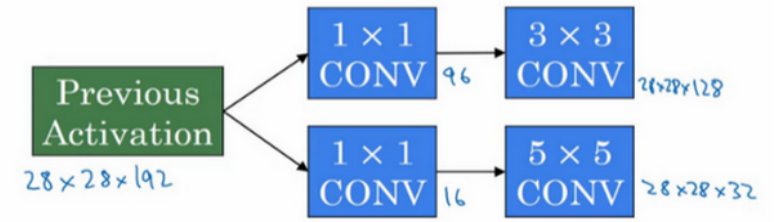
这就是 **Inception** 模块的主要思想，我们在这总结一下。下节课，我们将演示一个完整的 **Inception** 网络。

2.7 Inception 网络（Inception network）

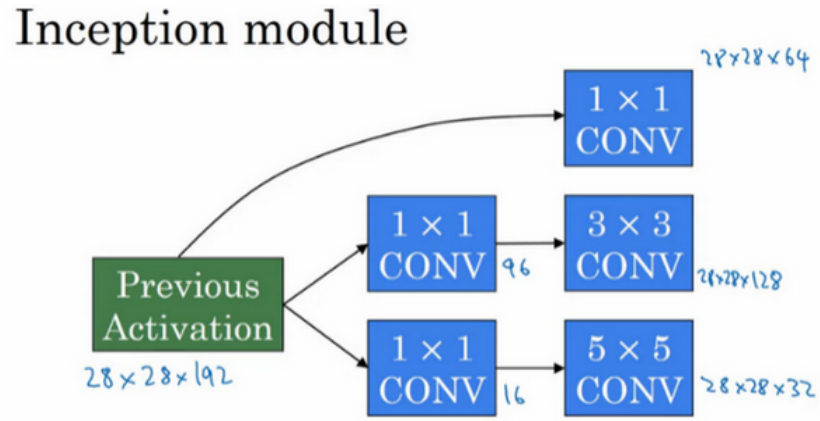
在上节视频中，你已经见到了所有的 **Inception** 网络基础模块。在本视频中，我们将学习如何将这些模块组合起来，构筑你自己的 **Inception** 网络。



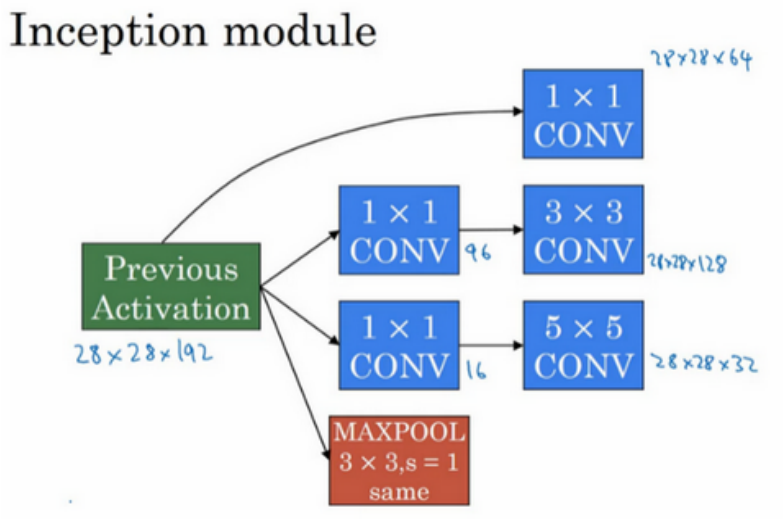
Inception 模块会将之前层的激活或者输出作为它的输入，作为前提，这是一个 28×28×192 的输入，和我们之前视频中的一样。我们详细分析过的例子是，先通过一个 1×1 的层，再通过一个 5×5 的层，1×1 的层可能有 16 个通道，而 5×5 的层输出为 28×28×32，共 32 个通道，这就是上个视频最后讲到的我们处理的例子。



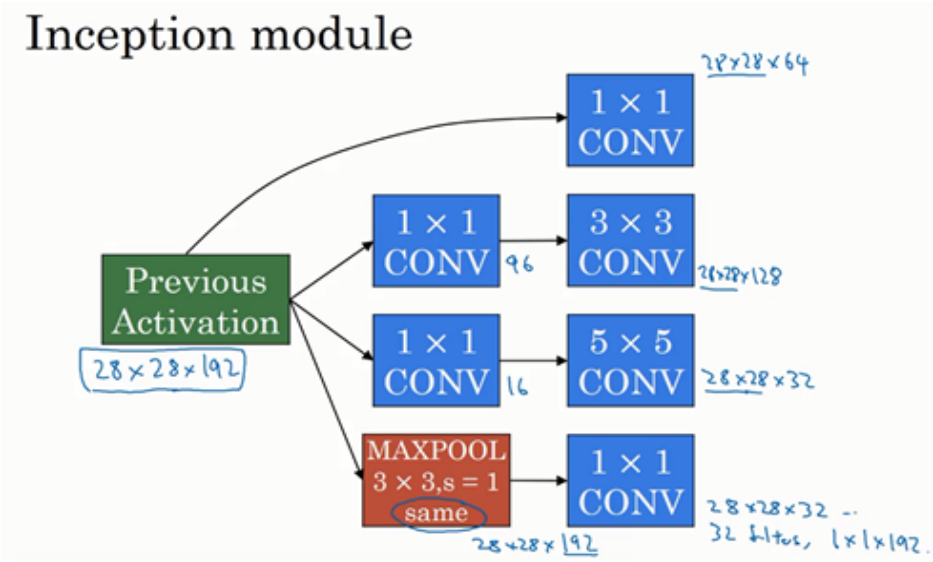
为了在这个 3×3 的卷积层中节省运算量，你也可以做相同的操作，这样的话 3×3 的层将会输出 28×28×128。



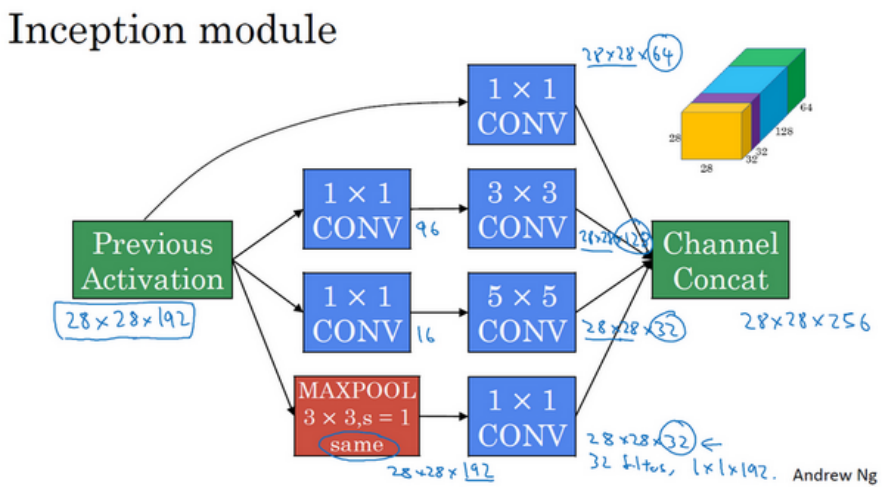
或许你还想将其直接通过一个 1×1 的卷积层，这时就不必在后面再跟一个 1×1 的层了，这样的话过程就只有一步，假设这个层的输出是 28×28×64。



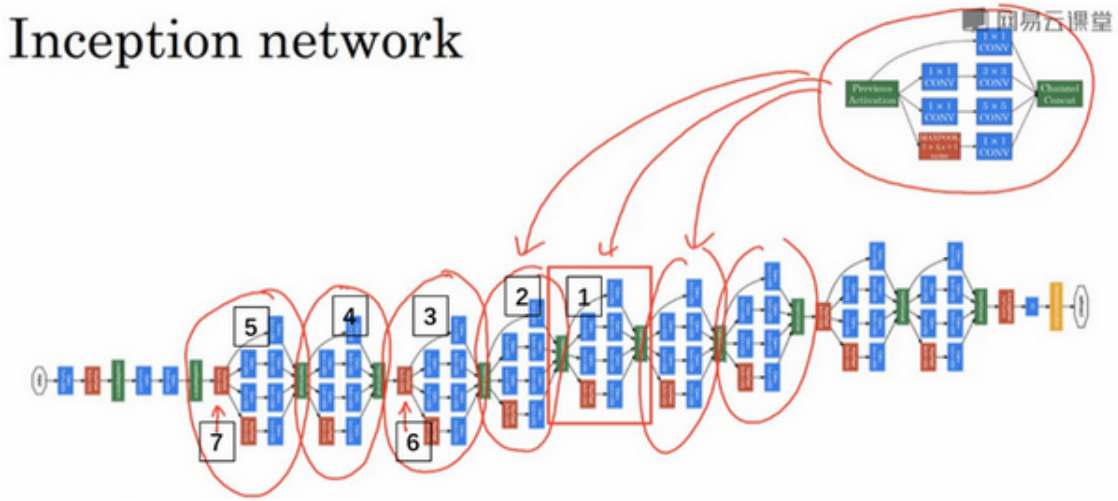
最后是池化层。



这里我们要做些有趣的事情，为了能在最后将这些输出都连接起来，我们会使用 **same** 类型的 **padding** 来池化，使得输出的高和宽依然是 28×28，这样才能将它与其他输出连接起来。但注意，如果你进行了最大池化，即便用了 **same padding**，3×3 的过滤器，**stride** 为 1，其输出将会是 28×28×192，其通道数或者说深度与这里的输入（通道数）相同。所以看起来它会有很多通道，我们实际要做的就是再加上一个 1×1 的卷积层，去进行我们在 1×1 卷积层的视频里所介绍的操作，将通道的数量缩小，缩小到 28×28×32。也就是使用 32 个维度为 1×1×192 的过滤器，所以输出的维度其通道数缩小为 32。这样就避免了最后输出时，池化层占据所有的通道。



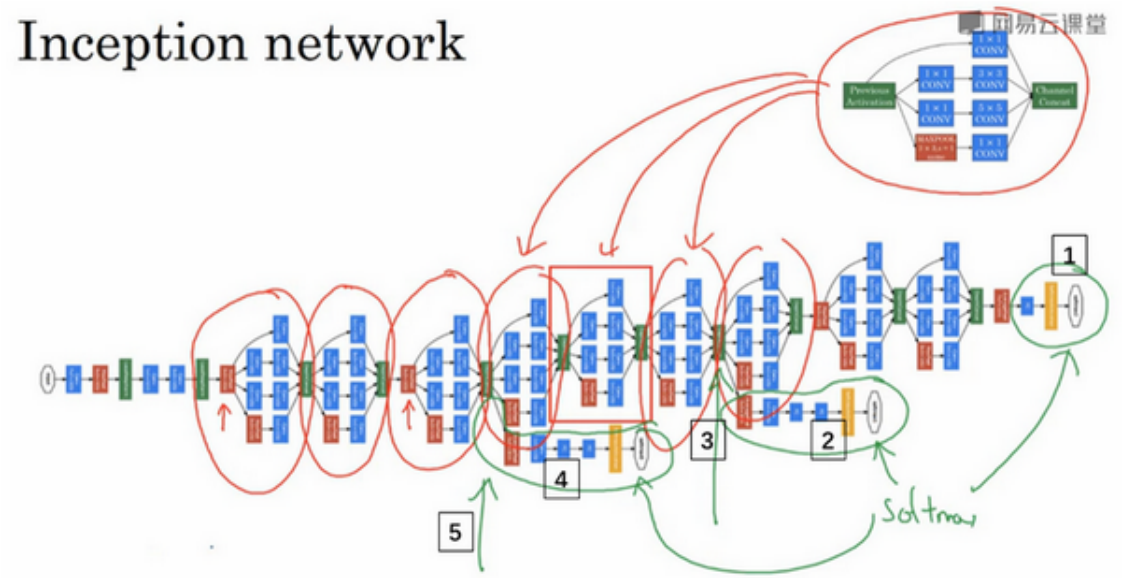
最后，将这些方块全都连接起来。在这过程中，把得到的各个层的通道都加起来，最后得到一个 28×28×256 的输出。通道连接实际就是之前视频中看到过的，把所有方块连接在一起的操作。这就是一个 **Inception** 模块，而 **Inception** 网络所做的就是将这些模块都组合到一起。



这是一张取自 **Szegety et al** 的论文中关于 **Inception** 网络的图片，你会发现图中有许多重复的模块，可能整张图看上去很复杂，但如果你只截取其中一个环节（编号 1），就会发现这是在前一页 **ppt** 中所见的 **Inception** 模块。

我们深入看看里边的一些细节，这是另一个 **Inception** 模块（编号 2），这也是一个 **Inception** 模块（编号 3）。这里有一些额外的最大池化层（编号 6）来修改高和宽的维度。这是另外一个 **Inception** 模块（编号 4），这是另外一个最大池化层（编号 7），它改变了高和宽。而这里又是另一个 **Inception** 模块（编号 5）。

所以 **Inception** 网络只是很多这些你学过的模块在不同的位置重复组成的网络，所以如果你理解了之前所学的 **Inception** 模块，你就也能理解 **Inception** 网络。



事实上，如果你读过论文的原文，你就会发现，这里其实还有一些分支，我现在把它们加上去。所以这些分支有什么用呢？在网络的最后几层，通常称为全连接层，在它之后是一个 **softmax** 层（编号 1）来做出预测，这些分支（编号 2）所做的就是通过隐藏层（编号 3）来做出预测，所以这其实是一个 **softmax** 输出（编号 2），这（编号 1）也是。这是另一条分支（编号 4），它也包含了一个隐藏层，通过一些全连接层，然后有一个 **softmax** 来预测，输出结果的标签。

你应该把它看做 **Inception** 网络的一个细节，它确保了即便是隐藏单元和中间层（编号 5）也参与了特征计算，它们也能预测图片的分类。它在 **Inception** 网络中，起到一种调整的效果，并且能防止网络发生过拟合。

还有这个特别的 **Inception** 网络是由 **Google** 公司的作者所研发的，它被叫做 **GoogleLeNet**，这个名字是为了向 **LeNet** 网络致敬。在之前的视频中你应该了解了 **LeNet** 网络。我觉得这样非常好，因为深度学习研究人员是如此重视协作，深度学习工作者对彼此的工作成果有一种强烈的敬意。

最后，有个有趣的事实，**Inception** 网络这个名字又是缘何而来呢？**Inception** 的论文特地提到了这个模因（**meme**，网络用语即“梗”），就是“我们需要走的更深”（**We need to go deeper**），论文还引用了这个网址（http://knowyourmeme.com/memes/we-need-to-go-deeper），连接到这幅图片上，如果你看过 **Inception**（**盗梦空间**）这个电影，你应该能看懂这个由来。作者其实是通过它来表明了建立更深的神经网络的决心，他们正是这样构建了 **Inception**。我想一般研究论文，通常不会引用网络流行模因（梗），但这里显然很合适。

http://knowyourmeme.com/memes/we-need-to-go-deeper Andrew Ng

最后总结一下，如果你理解了 **Inception** 模块，你就能理解 **Inception** 网络，无非是很多个 **Inception** 模块一环接一环，最后组成了网络。自从 **Inception** 模块诞生以来，经过研究者们的不断发展，衍生了许多新的版本。所以在你们看一些比较新的 **Inception** 算法的论文时，会发现人们使用这些新版本的算法效果也一样很好，比如 **Inception V2**、**V3** 以及 **V4**，还有一个版本引入了跳跃连接的方法，有时也会有特别好的效果。但所有的这些变体都建立在同一种基础的思想上，在之前的视频中你就已经学到过，就是把许多 **Inception** 模块通过某种方式连接到一起。通过这个视频，我想你应该能去阅读和理解这些 **Inception** 的论文，甚至是一些新版本的论文。

直到现在，你已经了解了许多专用的神经网络结构。在下节视频中，我将会告诉你们如何真正去使用这些算法来构建自己的计算机视觉系统，我们下节视频再见。