Rethinking the Inception Architecture for Computer Vision

摘要

卷积网络是计算机视觉多种任务的核心解决方案。从 2014 年开始，深度卷积网络开始成为主流，很多学者在此领域取得了实质性的进展。对很多任务来说通过增加模型的尺寸和计算消耗将其转化为质量收益（只要提供足够多的被标签的数据来训练），在移动端或者大数据的场景下，计算效率和较少的参数仍然是促进性的因素。在这里我们探索扩展网络的方法，目的是通过合理的因式化卷积和激烈的正则化尽可能高效地利用增加的计算力。我们在 ILSVRC2012 分类挑战的验证集上应用了我们的方法，它的结果比之前的都要好，对于使用了计算消耗为每次推断 5 千万次乘法加法和小于两千五百万的参数的单框架评估取得了 top-1 错误率 21.2%，top-5 错误率 5.6%，融合四个模型和多切片的评估，得到了 3.5%的 top-5 错误率和 17.3%的 top-1 错误率。

1 介绍

从 2012 年 Krizhevsky 等人赢得了 ImageNet 竞赛起，他们的网络“AlexNet”已经成功了应用到了许多计算机视觉任务中，例如目标检测，分割，行人姿势评估，视频分类，目标跟踪和超分辨率。

这些成功推动了一个新研究领域，这个领域主要专注于寻找更高效运行的卷积神经网络。从 2014 年开始，通过利用更深更宽的网络，网络架构的质量得到了明显改善。VGGNet 和 GoogLeNet 在 2014 ILSVRC 分类挑战上取得了类似的高性能。一个有趣的发现是在分类性能上的收益趋向于转换成各种应用领域上的显著质量收益。这意味着深度卷积架构上的架构改进可以用来改善大多数越来越多地依赖于高质量、可学习视觉特征的其它计算机视觉任务的性能。网络质量的改善也导致了卷积网络在新领域的应用，在 AlexNet 特征不能与手工精心设计的解决方案竞争的情况下，例如，检测时的候选区域生成。

尽管 VGGNet[18]具有架构简洁的强有力特性，但它的成本很高：评估网络需要大量的计算。另一方面，GoogLeNet 的 Inception 架构也被设计为在内存和计算预算严格限制的情况下也能表现良好。例如，GoogleNet 只使用了 500 万参数，与其前身 AlexNet 相比减少了 12 倍，AlexNet 使用了 6000 万参数。此外，VGGNet 使用了比 AlexNet 大约多 3 倍的参数。

Inception 的计算成本也远低于 VGGNet 或其更高性能的后继者。这使得可以在大数据场景中，在大量数据需要以合理成本处理的情况下或在内存或计算能力固有地受限情况下，利用 Inception 网络变得可行，例如在移动视觉设定中。通过应用针对内存使用的专门解决方案，或通过计算技巧优化某些操作的执行，可以减轻部分这些问题。但是这些方法增加了额外的复杂性。此外，这些方法也可以应用于优化 Inception 架构，再次扩大效率差距。

然而，Inception 架构的复杂性使得更难以对网络进行更改。如果单纯地放大架构，大部分的计算收益可能会立即丢失。此外，并没有提供关于导致 GoogLeNet 架构的各种设计决策的贡献因素的明确描述。这使得它更难以在适应新用例的同时保持其效率。例如，如果认为有必要增加一些 Inception 模型的能力，将滤波器组大小的数量加倍的简单变换将导致计算成本和参数数量增加 4 倍。这在许多实际情况下可能会被证明是禁止或不合理的，尤其是在相关收益适中的情况下。在本文中，我们从描述一些一般原则和优化思想开始，对于以有效的方式扩展卷积网络来说，这被证实是有用的。虽然我们的原则不局限于 Inception 类型的网络，但是在这种情况下，它们更容易观察，因为 Inception 类型构建块的通用结构足够灵活，可以自然地合并这些约束。这通过大量使用降维和 Inception 模块的并行结构来实现，这允许减轻结构变化对邻近组件的影响。但是，对于这样做需要谨慎，因为应该遵守一些指导原则来保持模型的高质量。

2 通用设计原则

这里我们将介绍一些具有卷积网络的、具有各种架构选择的、基于大规模实验的设计原则。在这一点上，以下原则的效用是推测性的，另外将来的实验证据将对于评估其准确性和有效领域是必要的。然而，严重偏移这些原则往往会导致网络质量的恶化，修正检测到的这些偏差状况通常会导致改进的架构。

1， 避免代表性的瓶颈，特别是在网络的前期。前馈网络可以由从输入层到分类器或者回归器的非周期性的图来表示。这为信息的流动定义了明确的方向。对于分离输入和输出的任何切口，我们可以估计通过这个切口的信息量。应该避免极端压缩的瓶颈。通常来说，在到达用于手边任务的最终表达层表示尺寸应该从输入到输出缓慢减小。理论上，信息容量不能仅仅以代表层的维度评估，因为它没有关注例如正确的结构这些因素。维度仅仅提供了一个信息容量的大概估计。

2， 更高的维度代表着更容易在网络中局部处理，在卷积网络中增加每个图块的活跃性允许更多的解开的特征。结果是网络会更快的训练。

3， 空间聚合可以在低维上嵌入，这样会有很少或几乎没有代表性力量的损失。例如，在执行更多展开（例如 3×3）卷积之前，可以在空间聚合之前减小输入表示的维度，没有预期的严重不利影响。我们假设，如果在空间聚合上下文中使用输出，则相邻单元之间的强相关性会导致维度缩减期间的信息损失少得多。鉴于这些信号应该易于压缩，因此尺寸减小甚至会促进更快的学习。

4， 平衡网络的宽度和深度。网络的出色性能表现可以通过平衡每一层滤波器的数量和网络的深度达到。增加网络的宽度和深度能够达到更好的网络质量，然而，如果两者并行增加，则可以达到恒定计算量的最佳改进。因此，计算预算应该在网络的深度和宽度之间以平衡方式进行分配。

尽管这些原则可能会有用，但并不是直接就可以用他们来提升网络质量，我们的意思是在模棱两可的情况下批判的使用他们。

3 分解大尺寸的卷积滤波器

GoogLeNet 的大部分原始收益来自于非常多的降维的使用，这可以被视为以计算有效的方式分解卷积的特例。例如说考虑一个 1*1 的卷积层后跟随着 3*3 的卷积层。在视觉网络中，邻近活跃度的输出值关联度很高。因此，我们可以预期，它们的激活可以在聚合之前被减少，并且这应该会导致类似的富有表现力的局部表示。

这里我们用不同的设置来探索分解卷积的其他方式。特别是为了增加解决方案的计算效率。因为 Inception 网络是全卷积，每激活一个权重对应于一个乘法

。因此任何计算消耗的减少导致参数的减少。这意味着使用合适的分解我们能够得到更多解开的参数，因此训练更快。我们也可以使用节省计算和内存的方法来增加网络的滤波器组的尺寸，同时保留在一个电脑上训练每个模型的能力。

3.1 分解为更小的滤波器

带有大滤波器的卷积层（比如说 5*5 或者 7*7）消耗了不合理的计算力。例如 n 个 5*5 的卷积层在一个有 m 个滤波器的网格上是 3*3 的滤波器消耗的计算能力的 3.78 倍。当然 5*5 的滤波器能捕捉到边界信号在更早的层里。所以滤波器尺寸的减小会带来巨大的损失。然而我们可以问 5*5 的卷积层可不可以被有着相同输入和输出深度同时参数更少的多层网络所代替呢。如果我们放大 5*5 卷积的计算图，我们看到每个输出看起来像一个小的完全连接的网络，在其输入上滑过 5*5 的块（见图 1）。由于我们正在构建视觉网络，所以通过两层的卷积结构再次利用平移不变性来代替全连接的组件似乎是很自然的：第一层是 3*3 卷

积，第二层是在第一层的 3*3 输出网格之上的一个全连接层（见图 1）。通过在输入激活网格上滑动这个小网络，用两层 3*3 卷积来替换 5*5 卷积（比较图 4 和 5）。

该设定通过相邻块之间共享权重明显减少了参数数量。为了分析预期的计算成本节省，我们将对典型的情况进行一些简单的假设：我们可以假设 n=αm，也就是我们想通过常数 α 因子来改变激活/单元的数量。由于 5×5 卷积是聚合的，α 通常比 1 略大（在 GoogLeNet 中大约是 1.5）。用两个层替换 5×5 层，似乎可以通过两个步骤来实现扩展：在两个步骤中通过根号 α 增加滤波器数量。为了简化我们的估计，通过选择 α=1（无扩展），如果我们单纯地滑动网络而不重新使用相邻网格图块之间的计算，我们将增加计算成本。滑动该网络可以由两个 3×3 的卷积层表示，其重用相邻图块之间的激活。这样，我们最终得到一个计算量减少到 9+9/25 的网络，通过这种分解导致 28％的相对增益。每个参数在每个单元的激活计算中只使用一次，所以参数计数具有完全相同的节约。不过，这个设置提出了两个一般性的问题：这种替换是否会导致任何表征力的丧失？如果我们的主要目标是对计算的线性部分进行分解，是不是建议在第一层保持线性激活？我们已经进行了几个控制实验（例如参见图 2），并且在分解的所有阶段中使用线性激活总是逊于使用修正线性单元。我们将这个收益归因于网络可以学习的增强的空间变化，特别是如果我们对输出激活进行批标准化。当对维度减小组件使用线性激活时，可以看到类似的效果。

3.2 分解为非对称卷积

上面的结果说明大于 3*3 的卷积核并不一定总是有用的因为他们总可以被分解为一组 3*3 的卷积。我们还可以问这样的问题，我们是否应该将他们分解的更小？例如说 2*2 的卷积。然而，我们找到了可以比 2*2 的卷积做的更好的，非对称的卷积。例如 n*1 的卷积。用 3*1 的卷积在后面跟着 1*3 的卷积等效于滑动一个以 3*3 的卷积相同的感受野的两层网络，如果输入和输出滤波器相同的话，对于相同输出数量的滤波器，两层的方案要便宜 33%。相比来说，将其分解成 2*2 的网络只节约了 11%的计算量。

理论上，我么可以更进一步去讨论将任何 n*n 的卷积替换成 1*n 和 n*1 的组合，这样节省的计算量会 n 倍增加，但是在实践中我们发现，这个分解在网络早期工作的不是很好，但是它在中等尺寸的网格中给出了很好的结果(在 m*m 特征图中，m∈[12,20])，在这个等级，使用 1*7 跟随 7*1 的卷积层能够得到很好的结果。

4 辅助分类器

[20]引入了辅助分类器的概念，以改善非常深的网络的收敛。最初的动机是将有用的梯度推向较低层，使其立即有用，并通过抵抗非常深的网络中的消失梯度问题来提高训练过程中的收敛。Lee 等人[11]也认为辅助分类器促进了更稳定的学习和更好的收敛。有趣的是，我们发现辅助分类器在训练早期并没有导致改善收敛：在两个模型达到高精度之前，有无侧边网络的训练进度看起来几乎相同。接近训练结束，辅助分支网络开始超越没有任何分支的网络的准确性，达到了更高的稳定水平。

另外，[20]在网络的不同阶段使用了两个侧分支。移除更下面的辅助分支对网络的最终质量没有任何不利影响。再加上前一段的观察结果，这意味着[20]最初的假设，这些分支有助于演变低级特征很可能是不适当的。相反，我们认为辅助分类器起着正则化项的作用。这是由于如果侧分支是批标准化的[7]或具有丢弃层，则网络的主分类器性能更好。这也为推测批标准化作为正则化项给出了一个弱支持证据。

5 有效的减少网格尺寸

传统的卷积网络使用一些池化操作来减少特征图的网格尺寸。为了避免代表性的瓶颈，在应用最大或者平均池化之前，需要扩展网络滤波器的维度。例如，从一个 k 个滤波器的

d*d 的网格开始。我们想要达到一个 2k 滤波器的(d/2)*(d/2)的网格，首先我们需要计算步进为一的 2k 个滤波器的卷积，然后应用额外的池化层。这意味着所有的计算耗费都来自于在较大的网格上进行 $2d^2k^2$ 次昂贵的卷积运算。另一种可能是先进行池化再进行卷积，这导致了计算耗费减少到了原来的四分之一。但是这产生了代表性的瓶颈因为全局代表维度跌到了 $(d/2)^2k$,这导致低价值的网络。为了代替这种做法，我们提出了另一个方法减少了计算量同时避免了代表性的瓶颈。我们可以用两个并行的步进为 2 的块：P 和 C。P 是池化层(最大池化或者平均池化)。他们的步进值都是 2.滤波器的连接方式如图 10 所示。

6 Inception-V2

这里我们联系上面的点提出一个新的结构来提高在 ILSVRC 2012 分类赛上的表现。我们网络的分布在表一给出。注意到我们把传统的 7*7 的网络基于 3.1 部分的思想分解成三个 3*3 的卷积，在 InceptionBUFEN 的网络，我们有是哪个传统的 Inception 模块每个有 288 个 35*35 的滤波器，我们使用在第五部分描述的网格减少技术将其减少到 288 个 17*17 的滤波器。紧跟其后的是在表格五中描述的 5 个分解 Inception 模型。使用在第五部分描述的网格减少部分将其减少到 8*8*1280，在粗糙的 8*8 的等级，我么你有两个 Inception 模型像在表 6 描述的那样，每个块连接的输出滤波器组的大小为 2048。网络结构的细节，包括 Inception 模块中的滤波器组的尺寸，在补充材料中给出，在提交的 tar 文件中的 model.txt 中给出。然而我们观察到只要符合第二部分的原则那么网络的质量就会相对稳定。尽管我们的网络有 42 层，我们的计算代价仅仅时 GoogLenet 的 2.5 倍，比 VGGnet 更加高效。

7 通过标签平滑来模型正则化

这里我们提出一个机制来通过估计训练中标签丢弃的排斥效应来正则化分类器层。

对于每个训练样本 x，我们的模型计算每个标签 k∈{1…K}：$p(k|x)=\frac{e^{z_k}}{\sum_{i=1}^{K} z_i}$的可能性。在这里$z_i$是对数过着非正则化的可能性。考虑到在训练样本中实际的标签 q(k|x)的分布，因此归一化后$\sum_k p(k|x) = 1$。为了简洁，我们省略 p 和 q 对样本 x 的依赖。我们将样本损失定义为交叉熵：$\ell=-\sum_{k=1}^{K}\log(p(k))q(k)$。最小化交叉熵等价于最大化标签对数似然期望，其中标签是根据它的实际分布 q(k)选择的。交叉熵损失对于 z_k 是可微的，因此可以用来进行深度模型的梯度训练。其梯度有一个更简单的形式：$\partial\ell/\partial z_k=p(k)-q(k)$，它的范围在−1 到 1 之间。考虑单个真实标签 y 的例子，对于所有 k≠y，有 q(y)=1，q(k)=0。在这种情况下，最小化交叉熵等价于最大化正确标签的对数似然。对于一个特定的样本 x，其标签为 y，对于 $q(k)=\delta_{k,y}$，最大化其对数概率，$\delta_{k,y}$ 为狄拉克 δ 函数，当且仅当 k=y 时， δ 函数值为 1，否则为 0。对于有限的 z_k，不能取得最大值，但对于所有 k≠y，如果 $z_y \gg z_k$——也就是说，如果对应实际标签的逻辑单元远大于其它的逻辑单元，那么对数概率会接近最大值。然而这可能会引起两个问题。首先，它可能导致过拟合：如果模型学习到对于每一个训练样本，分配所有概率到实际标签上，那么它不能保证泛化能力。第二，它鼓励最大的逻辑单元与所有其它逻辑单元之间的差距变大，与有界限的梯度$\partial\ell/\partial z_k$ 相结合，这会降低模型的适应能力。直观上讲这会发生，因为模型变得对它的预测过于自信。

我们提出了一个鼓励模型不那么自信的机制。如果目标是最大化训练标签的对数似然，这可能不是想要的，但它确实使模型正规化并使其更具适应性。这个方法很简单。考虑标签 u(k)的分布和平滑参数 ϵ，与训练样本 x 相互独立。对于一个真实标签为 y 的训练样本，我们用 $q'(k)=(1-\epsilon)\,\delta_{k,y}+\epsilon u(k)$来替代标签分布 $q(k|x)=\delta_{k,y}$. 其由最初的实际分布 q(k|x)和固定分布 u(k)混合得到，它们的权重分别为 1−ϵ 和 ϵ。这可以看作获得标签 k 的分布如下：首先，将其设置为真实标签 k=y；其次，用分布 u(k)中的采样和概率 ϵ 替代 k。我们建议使用标签

上的先验分布作为 u(k)。在我们的实验中，我们使用了均匀分布 u(k)=1/K，以便使得

$$q'(k)=(1-\epsilon)\delta_{k,y}+\epsilon K.$$

我们将真实标签分布中的这种变化称为标签平滑正则化，或 LSR。

注意，LSR 实现了期望的目标，阻止了最大的逻辑单元变得比其它的逻辑单元更大。实际上，如果发生这种情况，则一个 q(k)将接近 1，而所有其它的将会接近 0。这会导致 $q'(k)$ 有一个大的交叉熵，因为不同于 $q(k)=\delta_{k,y}$，所有的 $q'(k)$都有一个正的下界。

LSR 的另一种解释可以通过考虑交叉熵来获得：

$$H(q',p)=-\sum_{k=1}^{K}\log p(k)q'(k)=(1-\epsilon)H(q,p)+\epsilon H(u,p)$$

因此，LSR 等价于用一对这样的损失 H(q,p)和 H(u,p)来替换单个交叉熵损失 H(q,p)。第二个损失惩罚预测的标签分布 p 与先验 u 之间的偏差，其中相对权重为 ϵ1−ϵ。注意，由于 H(u,p)=DKL(u|p)+H(u)和 H(u)是固定的，因此这个偏差可以等价地被 KL 散度捕获。当 u 是均匀分布时，H(u,p)是度量预测分布 p 与均匀分布不同的程度，也可以通过负熵−H(p)来度量（但不等价）；我们还没有实验过这种方法。

在我们的 K=1000 类的 ImageNet 实验中，我们使用了 u(k)=1/1000 和 ϵ=0.1。对于 ILSVRC 2012，我们发现对于 top-1 错误率和 top-5 错误率，持续提高了大约 0.2%（见表 3）。

8 训练方法

我们在 TensorFlow[1]分布式机器学习系统上使用随机梯度方法训练了我们的网络，使用了 50 个副本，每个副本在一个 NVidia Kepler GPU 上运行，批处理大小为 32，100 个 epoch。我们之前的实验使用动量方法[19]，衰减值为 0.9，而我们最好的模型是用 RMSProp [21]实现的，衰减值为 0.9，ϵ=1.0。我们使用 0.045 的学习率，每两个 epoch 以 0.94 的指数速率衰减。此外，阈值为 2.0 的梯度裁剪[14]被发现对于稳定训练是有用的。使用随时间计算的运行参数的平均值来执行模型评估。

9 在低分辨率输入下的性能

视觉网络的典型用例是用于检测的后期分类，例如在 Multibox [4]上下文中。这包括分析在某个上下文中包含单个对象的相对较小的图像块。任务是确定图像块的中心部分是否对应某个对象，如果是，则确定该对象的类别。这个挑战的是对象往往比较小，分辨率低。这就提出了如何正确处理低分辨率输入的问题。

普遍的看法是，使用更高分辨率感受野的模型倾向于导致显著改进的识别性能。然而，区分第一层感受野分辨率增加的效果和较大的模型容量、计算量的效果是很重要的。如果我们只是改变输入的分辨率而不进一步调整模型，那么我们最终将使用计算上更便宜的模型来解决更困难的任务。当然，由于减少了计算量，这些解决方案很自然就出来了。为了做出准确的评估，模型需要分析模糊的提示，以便能够“幻化”细节。这在计算上是昂贵的。因此问题依然存在：如果计算量保持不变，更高的输入分辨率会有多少帮助。确保不断努力的一个简单方法是在较低分辨率输入的情况下减少前两层的步长，或者简单地移除网络的第一个池化层。

为了这个目的我们进行了以下三个实验：

步长为 2，大小为 299×299 的感受野和最大池化。

步长为 1，大小为 151×151 的感受野和最大池化。

步长为 1，大小为 79×79 的感受野和第一层之后没有池化。

所有三个网络具有几乎相同的计算成本。虽然第三个网络稍微便宜一些，但是池化层的成本是无足轻重的（在总成本的 1%以内）。在每种情况下，网络都进行了训练，直到收敛，并在 ImageNet ILSVRC 2012 分类基准数据集的验证集上衡量其质量。结果如表 2 所示。虽然

分辨率较低的网络需要更长时间去训练，但最终结果却与较高分辨率网络的质量相当接近。

但是，如果只是单纯地按照输入分辨率减少网络尺寸，那么网络的性能就会差得多。然而，这将是一个不公平的比较，因为我们将在比较困难的任务上比较一个便宜 16 倍的模型。

表 2 的这些结果也表明，有人可能会考虑在 R-CNN [5]的上下文中对更小的对象使用专用的高成本低分辨率网络。

10 实验结果和对照

表 3 显示了我们提出的体系结构（Inception-v2）识别性能的实验结果，架构如第 6 节所述。每个 Inception-v2 行显示了累积变化的结果，包括突出显示的新修改加上所有先前修改的结果。标签平滑是指在第 7 节中描述的方法。分解的 7×7 包括将第一个 7×7 卷积层分解成 3×3 卷积层序列的改变。BN-auxiliary 是指辅助分类器的全连接层也批标准化的版本，而不仅仅是卷积。我们将表 3 最后一行的模型称为 Inception-v3，并在多裁剪图像和组合设置中评估其性能。

我们所有的评估都在 ILSVRC-2012 验证集上的 48238 个非黑名单样本中完成，如[16]所示。我们也对所有 50000 个样本进行了评估，结果在 top-5 错误率中大约为 0.1%，在 top-1 错误率中大约为 0.2%。在本文即将出版的版本中，我们将在测试集上验证我们的组合结果，但是我们上一次对 BN-Inception 的春季测试[7]表明测试集和验证集错误趋于相关性很好。

11 总结

我们提供了几个设计原则来扩展卷积网络，并在 Inception 体系结构的背景下进行研究。这个指导可以导致高性能的视觉网络，与更简单、更单一的体系结构相比，它具有相对适中的计算成本。Inception-v3 的最高质量版本在 ILSVR 2012 分类上的单裁剪图像评估中达到了 21.2％的 top-1 错误率和 5.6％的 top-5 错误率，达到了新的水平。与 Ioffe 等[7]中描述的网络相比，这是通过增加相对适中（2.5/times）的计算成本来实现的。尽管如此，我们的解决方案所使用的计算量比基于更密集网络公布的最佳结果要少得多：我们的模型比 He 等[6]的结果更好——将 top-5(top-1)的错误率相对分别减少了 25%(14%)，然而在计算代价上便宜了六倍，并且使用了至少减少了五倍的参数（估计值）。我们的四个 Inception-v3 模型的组合效果达到了 3.5％，多裁剪图像评估达到了 3.5\％的 top-5 的错误率，这相当于比最佳发布的结果减少了 25％以上，几乎是 ILSVRC 2014 的冠军 GoogLeNet 组合错误率的一半。

我们还表明，可以通过感受野分辨率为 79×79 的感受野取得高质量的结果。这可能证明在检测相对较小物体的系统中是有用的。我们已经研究了在神经网络中如何分解卷积和积极降维可以导致计算成本相对较低的网络，同时保持高质量。较低的参数数量、额外的正则化、批标准化的辅助分类器和标签平滑的组合允许在相对适中大小的训练集上训练高质量的网络。