背景

设计轻量化网络模型有很多种办法，比如剪枝、量化、知识蒸馏等，这些模型压缩的方法性能通常取决于给定的预训练模型；还有一些网络是着眼于设计高效的网络结构，比如 MobileNets，ShuffleNets 等，虽然这些模型在很少的 FLOPs 下获得了良好的性能，但特征映射之间的相关性和冗余性并未得到很好的利用。而一些高准确率的网络模型，其成功的关键因素之一——冗余的特征图，往往被研究者所忽视。而本文将从特征图的冗余性展开研究。

深度卷积神经网络通常使用了大量卷积，导致计算量很大。尽管最近的轻量化网络模型，MobileNet 和 ShuffleNet 引入了深度方向的卷积或通道重排操作，以使用较小的卷积核来构建有效的模型，但是 1×1 卷积层仍将占用大量内存和 FLOPs。

方法

一个训练好的深度神经网络其提取到的丰富甚至冗余的特征信息保证了模型对输入图片的全面理解。例如，图 1 展示了 ResNet-50 网络中，将经过第一个残差块后的特征图可视化，可以发现其中存在许多相似的特征图对，这就是神经网络中存在的特征图冗余的情况。而现在我们要做的不是考虑如何避免特征冗余，而是以一种简单有效的方式，生成更多相似的特征图。

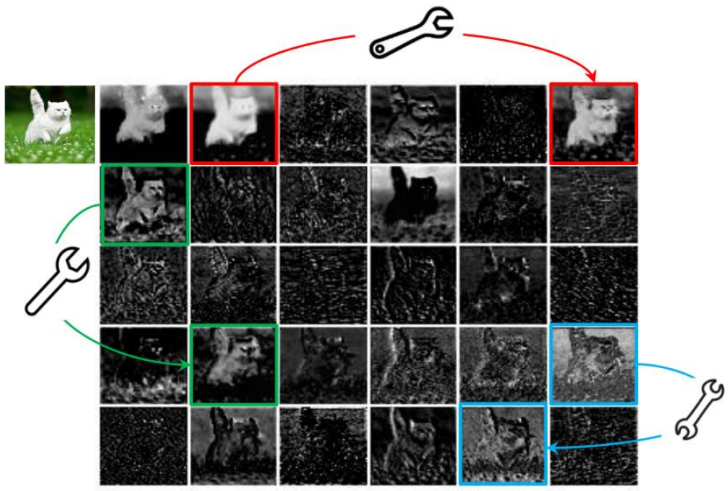

图 1

Ghost 模块

图 2 是普通卷积过程，由于卷积核数量和输入通道数通常非常大（例如 256 或 512），所需的 FLOPs 数量也非常大。

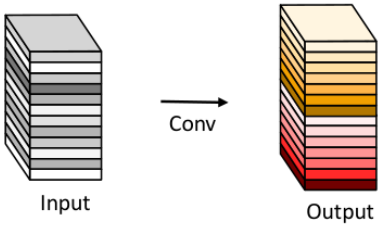

图 2

图 1 中所观察到的，卷积层的输出特征图通常包含很多冗余特征图。那么没有必要使用大量的 FLOPs 和参数来生成这些冗余的特征图，可以用简单的线性变换，以很小的成本生成许多能从原始特征发掘所需信息的“幻影”特征图。其过程示意图如下图所示：

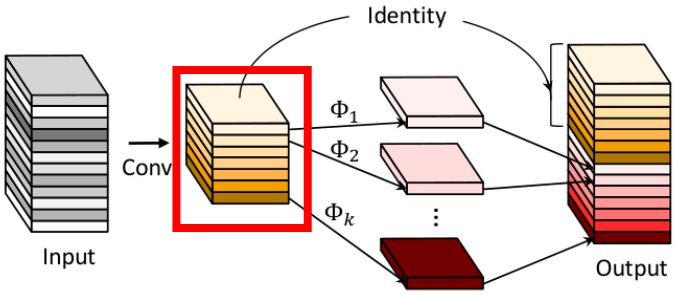

图 3

首先由普通卷积生成不含冗余的特征图 m 个，用 y_i' 表示（图 3 中的红色框），为了得到期望中的 n 个特征图，再将 y_i' 经过线性映射 $\Phi_{i,j}(y_i')$，其中 $i=1,\cdots,m$，$j=1,\cdots,s$，每个线性映射的平均内核大小为 $d\times d$。并且在这个线性映射中，每一个 y_i' 做最后一次映射时是进行恒等映射变换，最终得到 $n=m\times s$ 个特征图。

线性运算 Φ 在每个通道都进行运算，其计算量比普通卷积少得多。实际上，Ghost 模块中可能有几种不同的线性运算，例如 3×3 卷积和 5×5 卷积。理想情况下，$m\cdot(s-1)$ 个线性运算可以具有不同的形状和参数，但是特别是考虑到 CPU 或 GPU 的实用性，在线推理会受到阻碍。因此，为了使网络更高效，建议在一

个 Ghost 模块中采用相同大小的线性运算。

至此可以发现 Ghost 模块具有两个超参数，也就是，s 用于生成 m 个内在特征图，以及用于计算“幻影”特征图的线性运算的 $d \times d$ 卷积核。

作者固定参数 s，将其设置为 2，调整 d，下表是在 CIFAR-10 验证集上的结果。可以看到，当 $d=3$ 的时候，Ghost 模块的性能优于其他的 Ghost 模块。这是因为当卷积核大小为 1×1 时无法在引入空间信息，而较大的内核例如 5×5 或者 7×7 会导致过拟合和更多计算。因此，选用 $d=3$ 来提高效率。

表 1

d	Weights (M)	FLOPs (M)	Acc. (%)
VGG-16	15.0	313	93.6
1	7.6	157	93.5
3	7.7	158	93.7
5	7.7	160	93.4
7	7.7	163	93.1

从表 2 中的结果可以看出，当作者增加 s 时，FLOPs 显着减少，并且准确性逐渐降低，当 $s=2$ 时，也就是将 VGG-16 压缩 2 倍时，Ghost 模块的性能比原始 VGG-16 模型准确率略高。

表 2

s	Weights (M)	FLOPs (M)	Acc. (%)
VGG-16	15.0	313	93.6
2	7.7	158	93.7
3	5.2	107	93.4
4	4.0	80	93.0
5	3.3	65	92.9

搭建 GhostNet

Ghost 瓶颈结构（图 4 所示）主要由两个堆叠的 Ghost 模块组成。

第一个 Ghost 模块用于增加通道数。这里将输出通道数与输入通道数之比称为扩展比；第二个 Ghost 模块减少通道数，以方便最后与跳跃连接的路径进行相加。然后，使用跳跃连接这两个 Ghost 模块的输入和输出。注意，这里第二个 Ghost 模块之后不使用 ReLU，是借鉴了 MobileNetV2。对于步长为 2 的情况，跳跃连接由下采样层和步长为 2 的深度方向的卷积来实现。

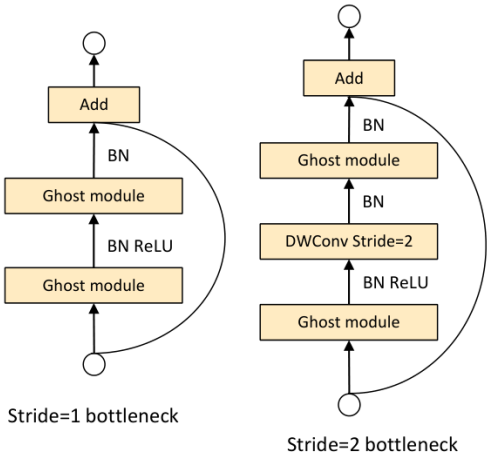

图 4

总结

个人对这篇文章的理解就是将深度分离卷积颠倒一下，先进行逐点卷积，然后再进行深度方向的卷积，只不过再深度方向的卷积里增加了一个恒等映射变换。

疑问

这里提到的线性变换看起来还是卷积矩阵相乘操作，为什么能减少 FLOPs 呢？