卷积神经网络在移动平台上的性能优化

Tianyu Li

Beihang University, No. 37 Xueyuan Road, Haidian District, Beijing, P.R. China

摘要 本文将针对如何构建更高效的卷积神经网络从而在一些终端上取得更好的效果来进行研究。从卷积网络结构本身以及网络与一些神经网络加速模块的适应性，以及量化模式等角度进行深入探讨。

Keywords: 计算机视觉 · 卷积神经网络 · 量化

1 Introduction

随着深度学习技术的迅猛发展以及其在生活中的广泛使用，我们迫切需要提升神经网络的性能。提高深度神经网络性能的最直接方法是增加它们的大小。这包括增加网络的层数以及其宽度（每一层的单元或 channels 数量）。这是一种简单有效获得更好模型的方法，尤其是在如今训练数据日渐增多的情况下。这样的模型的拟合能力很强，因此在计算机视觉领域中，如 ImageNet 这样的测试集的准确率越来越高。但这些较大的模型在移动设备或一些其它终端上往往需要较长的时间才可以完成一次预测，而安防等情景上对低延时的需求是十分必要的。因此像海思这样的公司在自己的端上计算模块中加入了神经网络加速模块，其设计针对卷积神经网络等流行计算模式进行优化，使其在低成本情况下获得了最好的计算性能。这样的神经网络加速模块远超 CPU 的神经网络计算能力使一些较大模型在终端上的应用成为了可能。同时，我们考虑到终端对内存优化的较高要求，需要去构建参数数量很少，运算次数较低的模型，同时又保留其同水平的分类性能。因此我们在网络结构上的优化做了很多研究。

单纯增加深度神经网络大小的方式有很多弊端。首先，更大的模型意味着更多的参数数量，这会在很多情境下导致过拟合，尤其是训练数据不足的情况下。考虑到训练数据的增加会耗费极大的成本，包括数据采集，数据标注和数据清洗等一系列流程，这会成为影响模型性能的主要瓶颈。而更大的模型也意味着更多的浮点运算次数 (FLOPs)，这会导致更长的训练时间，从而使模型调整的难度增加。如果增加的 channels 并没有被激活（如大多数

权重最终接近于零），则浪费了大量计算。由于在应用中的计算资源总是有限的，因此即使主要目标是提高结果质量，计算资源的有效分配也十分重要，不能盲目地增加模型大小。在 ILSVRC 2014 中获胜的 GoogLeNet [10]要比在两年前获胜的 Krizhevsky et al. [7]有了 12× 的参数缩减。这意味着随着深度学习技术的发展，人们不再单调地去追求更大更深的网络。在某些领域，比如物体检测，使用更大的模型或者更深的网络并不会带来最大的收益，反而应借助深层网络和经典计算机视觉领域方法的协同作用，如 Girshick 等人的 R-CNN 算法 [3]。

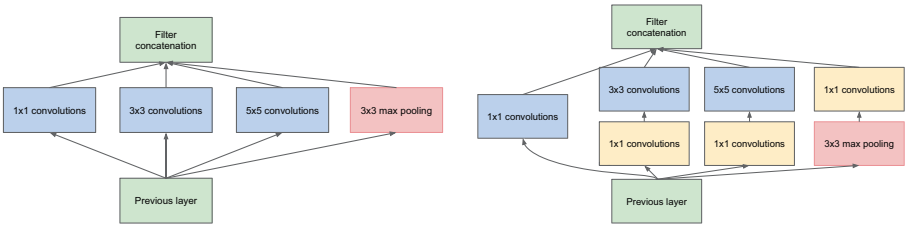
2 Network Architecture 网络结构

2.1 Inception

Incpetion [10]提出了解决该问题的根本方法是将神经网络完全连接的结构完全转化为稀疏连接，不论是在全连接层还是卷积层。Arora et al. [1]的研究指出，如果数据集的概率分布可以通过一个大的且非常稀疏的深度神经网络来表示，则可以通过分析最后一层激活的相关统计来逐层构建最优网络拓扑。从而聚类具有高度相关相关性输出的神经元。尽管严格的数学证明需要非常苛刻的条件限制，但这一陈述与众所周知的 Hebbian 原理“neurons that fire together, wire together”有一定相通之处，这表明了在实践中即使在一些较为普通的情况，这样的基本思想也适用。

但是，当涉及非均匀稀疏数据结构的数值计算时，目前的计算设备的性能会受到极大的影响。即使算术运算的数量减少了一百倍，稀疏的结构会导致 Cache 无法命中，此时访存的开销会十分重要，以至于构建这样的稀疏矩阵也无法获得可见的性能提升 [7,9]。结构的均匀性和大量滤波器以及更大的批大小可以对如今的密集计算设备有更好的优化。因此 Inception 致力于找到一些最优稀疏矩阵的网络拓扑的密集型结构表示来解决这一问题。

Inception 为了达到视觉上的 *translation invariance*，即平移不变性，使用卷积构建块结构并在空间上重复它。其认为在网络靠前的单元中，每个单元对应了一个输入图像的一个区域，也就是说每个单元作为一个滤波器与特定的图像对应。而更靠近输入的那些层中，相关单元的位置都集中在图像的局部区域。因此，大量的单元集中在同一个区域中，需要在下一层使用 1×1 卷积来覆盖它们。另外，在更大的区域里可能也有一些单元的集中，因此需要使用更大的卷积核来覆盖这些区域。如图1(a)，Inception 为了方便，在结



(a) Inception module, naïve version (b) Inception module with dimension reductions

图 1. Inception module

构中使用了 1×1，3×3，5×5 的卷积核，它们的输出拼接成单个向量，再连接到下一层的输入端。此外，Inception 使用了并行的池化路径在每一个构建块结构中，旨在提升模型表现。

但是这样的网络结构会导致 3×3 和 5×5 卷积核被连接到高维端，从而导致运算次数的爆炸式增加。因此在它们之前加入了 1×1 用以压缩数据，虽然被压缩的数据会更难以被建模。另外，1×1 要使用线性的激活函数来整合数据。因此，Inception 以其可以在任意阶段显著增加模型的单元数量，而防止其运算复杂度不受控制地爆炸式增加。因为大量滤波器的输出在接触到如 5×5 卷积这样的运算之前就已经被减小。

2.2 ResNet

ResNet [5]提出了 Identity Mapping by Shortcuts 和 Deeper Bottleneck Architectures 两种使更深的网络更有效的处理方法。前者是使用 *shortcut* 手段来解决梯度丢失的问题，后者则利用 *bottleneck* 结构来使模型更小更高效。

Identity Mapping by Shortcuts. Shortcuts 在网络更深时，通过增加一个残差分支来传导梯度，防止梯度在过多的卷积层的传导中丢失。这使得在更深的网络中，增加深度保证了性能不会降低。文中对比了几种 *shortcuts* 的方法：projection，zero padding 和 identity。

Identity.

$$y = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \tag{1}$$

等式 (1) 代表一个 building block，其中的 $\mathcal{F}$ 代表一个残差函数，往往由两层 3×3 卷积层构建而成，如图 2.

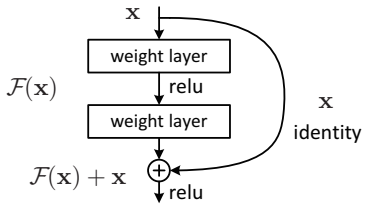


图 2. Residual learning: a building block.

Projection.

$$y = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \tag{2}$$

等式 (2) 中的 W_s 为一个线性的投影函数，一般使用 1×1 卷积层来实现。但这会增加模型的参数量和时间复杂度，且实验证明在一般情况下，它并没有提升的模型性能。

相比来说，Identity (1) 不会增加模型的参数和计算的复杂度，因此在一般状况下，Shortcut 分支直接使用 Identity 是比 Projection 更优的选择。而实际应用中应用中 Projection 用于当 Shortcut 分支的输入和输出的 dimensions 不同时，使用 1×1 卷积层进行维度的统一。

Zero-padding. Zero-padding 也是在 Shortcut 分支维度不同时的一种统一维度的方法，它直接在增加的维度上填充 0 来匹配输出，但它的效果并不如 Projection 好。

Deeper Bottleneck Architectures. 我们需要更深的网络来获得更好的非线性表达能力，则网络的训练时间和在移动设备上的 inference 或者说 forward 时间是必须要考虑的因素。

因此，ResNet [5]提出了 *bottleneck* 设计来替换原来的 building block。如图3，对于每个残差函数 $\mathcal{F}$，我们使用了三个卷积层来代替原有的两层，分别是 1×1，3×3 和 1×1 卷积，其中 1×1 层负责减小然后增加（恢复）维度，使 3×3 层作为 bottleneck，可以具有较小的输入和输出，大大降低了模型的时间复杂度。

无参数的 identity shortcut 对于 bottleneck 架构尤为重要。如果用 projection 替换 bottleneck 中的 shortcut 分支，会导致时间复杂度和模型大小翻倍，因为 shortcut 连接到两个 1×1 卷积层的高维端。因此，identity shortcut 可以为 bottleneck 设计带来更高效的网络结构。

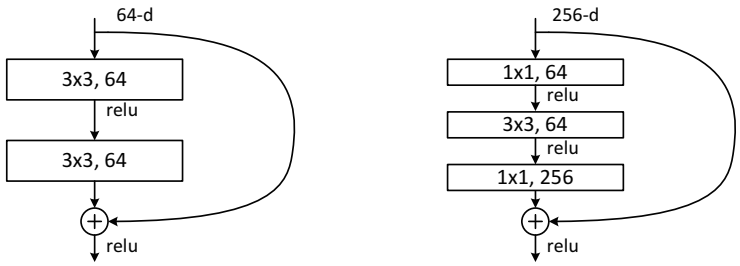


图 3. A deeper residual function $\mathcal{F}$

2.3 Inception V3, Xception

Inception 体系结构提出之后，由于其在各个领域优越的性能，优于 VGGNet [8]的分类性能表现，甚至优于其增强版 [4]，以及与 VGGNet 相比较少的参数数量，Inception 体系结构在业界被广泛使用。但对 Inception 结构过于鲁莽的改进，会导致其计算性能的性价比迅速地丢失，比如加倍滤波器的 channels 数量，会直接带来四倍的计算量提升，这在大多数有性能限制的应用场景中是不可接受的。同时，由于 [10]并没有详细地描述其结构的设计决策的清晰描述，这导致其他人对 Inception 的改动更加难以施行。因此 Inception V3 [11]提供了一些对 Inception 结构拓展的可行方法与思路。

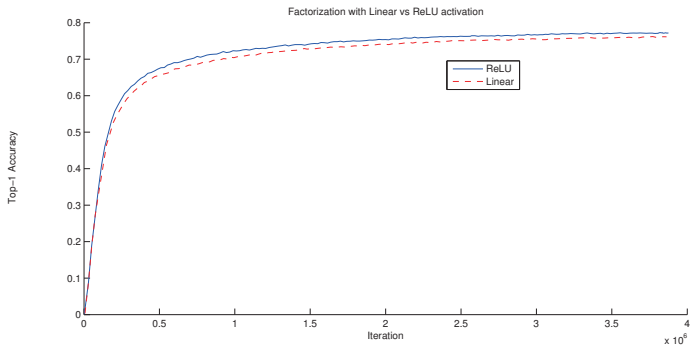


图 4. One of several control experiments between two Inception models, one of them uses factorization into linear + ReLU layers, the other uses two ReLU layers. After 3.86 million operations, the former settles at 76.2%, while the latter reaches 77.2% top-1 Accuracy on the validation set.

其认为应尽量减少 bottleneck 结构的使用，尤其是在网络早期。这是由于前馈网络可以表示成从输入层到分类器或回归器的非循环图，其定义了信息流的明确方向，但 bottleneck 结构会将信息流切割，并限制信息向后的流通。此外，对高维度数据的复杂表示可以移到网络内部，这可以通过激活更多的卷积层来实现对高维数据的有效表达。并且，网络可以通过尽快降低数据的维度，在低维度的情况下来完成数据在空间上的聚合，这是因为数据在空间中的关系信息可以被压缩，在低维情况下仍然可以被很好的表达。这样就可以更快地训练网络，因为大量的计算任务被分配在了低维数据中，这可以使模型有更低的 FLOPs。在增加网络大小时，要同时增加网络的宽度和深度。当这二者达到一个较好的平衡时，网络的性能会趋近于最佳。

由于大尺寸滤波器，如 5×5, 7×7 卷积核是如此地耗费计算资源，Inception V3 提供了一系列将大尺寸滤波器转化为多个小尺寸滤波器的思路。一个比较简明而有代表性的操作是，将 5×5 卷积转化为两个 3×3 卷积，这会获得 28% 的性能增益。且虽然两个 3×3 卷积是 5×5 的一种线性表达形式，在两个卷积核之间理应使用线性激活函数，但实验表明 (图4)，使用 ReLU 函数总是会获得更好的效果，尤其是加入了批标准化 [6]之后。

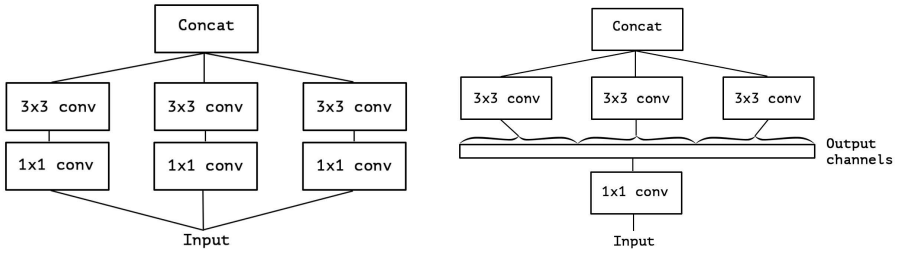


图 5. Left: A simplified Inception module. Right: A strictly equivalent reformulation of the simplified Inception module.

在 Inception V3 [11]的经典结构中，如图5(Left)，一个构建模块先由 1×1 卷积核和 average pooling 来将输入的 channel 映射到多个分支，随后数据经过 bottleneck 结构卷积核并通过 concat 操作组成输出端。

如果我们加以简化，使用 1×1 卷积核做分离操作，1×1 卷积核的 channels 分为多组，每组分别与一个 3×3 卷积核相连，如图5(Right)。这样大大减少了参数数量，因为每个 3×3 卷积核对应的 channel 数大大缩减。

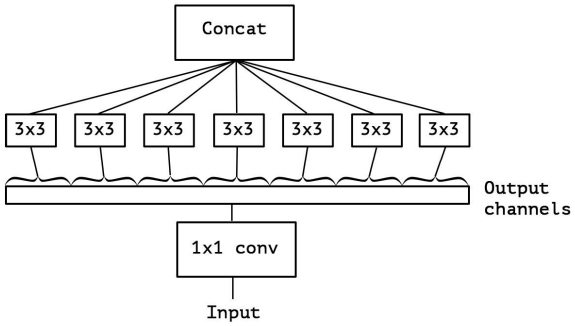


图 6. An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

如果将这样的思想极限化，每一个 1×1 卷积输出的 channel 都与一个 3×3 卷积相连，即每组的大小为 1，就可以得到一个极致化的 Inception 模块，如图6。这里使用 1×1 卷积提取 channels 之间的特征，并将其分离开，之后再由 3×3 卷积学习分离开的空间上的特征。这种模块在深度学习框架中被称作 channel-wise。

这样的模块给予了 Xception [2]很多启发。在 Xception 的被称作深度可分离卷积的基础结构中，首先使用 depthwise 卷积将输入的每一个 channel 分别使用一个空间上的卷积，之后使用被称作 pointwise 的 1x1 卷积核，将这些 channels 映射到深度可分离卷积的输出 channels 上。这一点与极致化的 Inception 模块有所区别，后者是先使用 1×1 卷积核分离各个 channel，再使用空间上的卷积来学习其它特征。这样的模块在深度学习框架中被称作 separable convolution。

值得注意的是，在每个深度可分离卷积中，激活函数 ReLU 应放在 1×1 卷积层之前，而在深度可分离卷积层之后，应加上批标准化。激活函数前置是深度可分离卷积与极致化的 Inception 模块的重要区别之一，后者每个卷积层后都会加上非线性激活函数。另外，由多个深度可分离卷积组成的一个 Xception module 使用了 shortcut connection 来增强模型表现。

参考文献

1. Arora, S., Bhaskara, A., Ge, R., Ma, T.: Provable bounds for learning some deep representations. In: ICML (2014) 2

2. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1800–1807 (2017) 7
3. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014) 2
4. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015) 5
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2016) 3, 4
6. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015) 6
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012) 2
8. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 5
9. Song, F., Dongarra, J.: Scaling up matrix computations on shared-memory manycore systems with 1000 cpu cores. In: Proceedings of the 28th ACM international conference on Supercomputing. pp. 333–342. ACM (2014) 2
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1–9 (2015) 2, 5
11. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2818–2826 (2016) 5, 6