前言

最近笔者在阅读关于骨骼点数据动作识别的文献Shift-GCN[2]的时候，发现了原来还有Shift卷积算子[1]这种东西，该算子是一种可供作为空间卷积的替代品，其理论上不需要增添额外的计算量和参数量，是一种做紧致模型设计的好工具。本文作为笔记纪录笔者的论文阅读思考，**如有谬误请联系指出，转载请联系作者并注明出处，谢谢**。

∇ 联系方式：

e-mail: FesianXu@gmail.com

QQ: 973926198

github: https://github.com/FesianXu

知乎专栏: 计算机视觉/计算机图形理论与应用

卷积计算及其优化

为了讨论的连续性，我们先简单回顾下传统的深度学习卷积计算。给定一个输入张量，如Fig 1.1中的蓝色块所示，其尺寸为$\mathbf{F} \in \mathbb{R}^{D_F \times D_F \times M}$；给定卷积核$\mathbf{K} \in \mathbb{R}^{D_K \times D_K \times M \times N}$，如Fig 1.1中的蓝色虚线框所示，为了方便起见，假定步进`stride = 1`，那么最终得到输出结果为$\mathbf{G} \in \mathbb{R}^{D_F \times D_F \times N}$，计算过程如式子(1.1)所示：

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} F_{k+\hat{i},l+\hat{j},m} \tag{1.1}$$

其中(k,l)为卷积中心，而$\hat{i} = i - \lfloor D_K/2 \rfloor$，$\hat{j} = j - \lfloor D_K/2 \rfloor$是卷积计算半径的索引。不难知道，该卷积操作的参数量为$M \times N \times D_K^2$。计算量也容易计算，考虑到每个卷积操作都需要对每个卷积核中的参数进行乘法计算，那么有乘法因子D_K^2，而考虑到`stride=1`而且存在填充，那么容易知道计算量为$M \times N \times D_F^2 \times D_K^2$ FLOPs。容易发现，卷积的计算量和参数量与卷积核大小D_K呈现着二次增长的关系，这使得卷积的计算量和参数量增长都随着网络设计的加深变得难以控制。

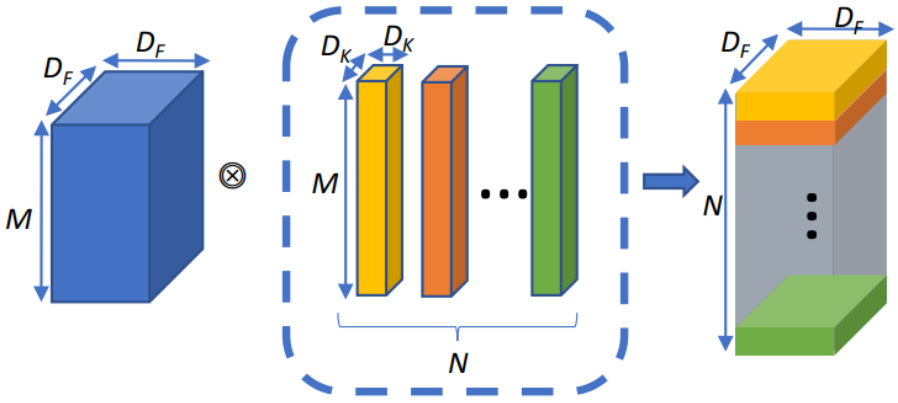

Fig 1.1 经典的卷积操作示意图。

在进一步对传统卷积计算进行优化之前，我们先分析一下卷积计算到底提取了什么类型的信息。以二维卷积为例子，卷积计算主要在两个维度提取信息，空间域和通道域，不过从本质上说，通道域的信息可以看成是原始输入（比如RGB图片输入）的层次化特征/信息的层叠，因此本质上二维卷积还是提取空间域信息，只不过在层叠卷积过程中，使得空间域信息按照层次的特点，散布在了通道域中。

知道了这一点，我们就可以把卷积过程中的空间卷积和通道卷积分离开了，从而得到了所谓的 **通道可分离卷积**[4,5]。如Fig 1.2所示，这类型的卷积将空间域和通道域卷积完全分开，在第一步只考虑空间域卷积，因此对于每个输入张量的通道，只会有唯一一个对应的卷积核进行卷积。数学表达为：

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} F_{k+\hat{i},l+\hat{j},m} \tag{1.2}$$

对比式子(1.1)和(1.2)，我们发现区别在于对卷积核的索引上，通过式子(1.2)输出的张量形状为$\hat{\mathbf{G}} \in \mathbb{R}^{D_F \times D_F \times M}$，为了接下来在通道域进行卷积，需要进一步应用1x1卷积，将通道数从M变为N，如式子(1.3)所示。

$$G_{k,l,n} = \sum_{m} P_{m,n} \hat{G}_{k,l,m} \tag{1.3}$$

其中$P \in \mathbb{R}^{M \times N}$为1x1卷积核。通道可分离卷积就是将传统卷积(1.1)分解为了(1.2)(1.3)两个步骤。

通过这种优化，可以知道卷积核参数量变为$M \times D_K^2$，而计算量变为$M \times D_K^2 \times D_F^2$ FLOPs。虽然理论上，深度可分离网络的确减少了计算量和参数量，但是实际上，因为深度可分离网络的实现使得访存[1]（memory access）过程占据了主导，使得实际计算占用率过小，限制了硬件的并行计算能力。

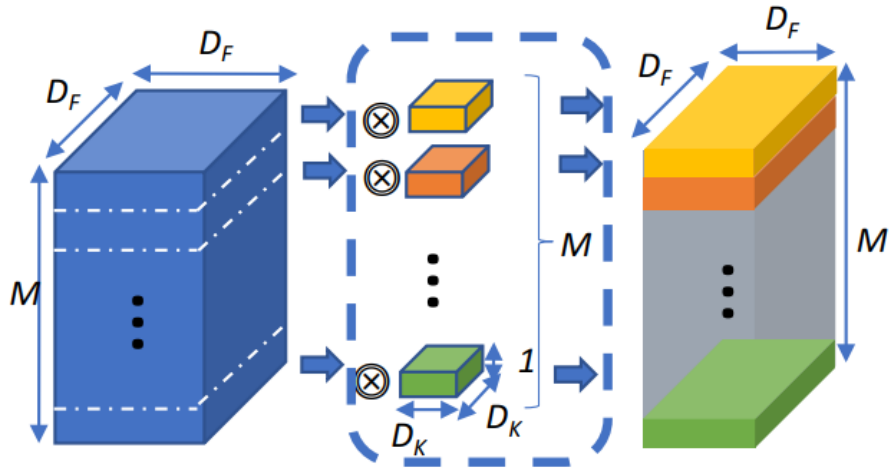

Fig 1.2 深度可分离卷积，对于输入张量的每一个通道，都有其专有的卷积核进行卷积，最后通过一个**1x1**的卷积即可完成通道数量的放缩。

我们用传统卷积和深度可分离卷积的`计算/访存`系数进行比较（仅考虑最基本的访存，即是将每个操作数都从内存中获取，而不考虑由于局部性原理[6]，而对重复的操作数进行访问导致的消耗）：

$$\text{传统卷积：}\frac{M\times N\times D_F^2\times D_K^2}{D_F^2\times(M+N)+D_F^2\times M\times N} \tag{1.4}$$

$$\text{深度可分离卷积：}\frac{M\times D_F^2\times D_K^2}{D_F^2\times 2M+D_K^2\times M} \tag{1.5}$$

式子(1.4)和(1.5)的比较最终会化简为比较$(M+N)/N$和$2M$的大小，越小意味着计算效率越高。我们发现，传统的卷积反而比深度可分离卷积的计算效率高得多。这是不利于程序并行计算的。

为此，文章[1]提出了Shift卷积算子，尝试解决这种问题。

Shift卷积算子

在Shift卷积算子中，其基本思路也是类似于深度可分离卷积的设计，将卷积分为空间域和通道域的卷积，通道域的卷积同样是通过1x1卷积实现的，而在空间域卷积中，引入了shift操作。我们接下来会详细地探讨shift操作的设计启发，细节和推导。

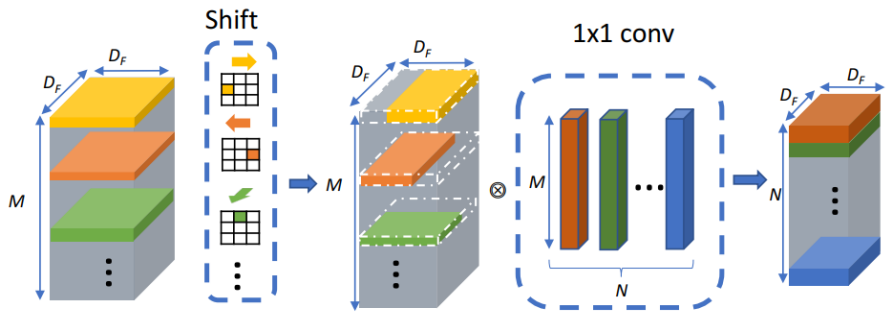

Fig 2.1 基于**Shift**的卷积可以分为**Shift**卷积算子和**1x1**卷积操作。

shift卷积算子的数学形式表达如式子(2.1)所示，如图Fig 2.1所示，shift卷积的每一个卷积核都是一个“独热”的算子，其卷积核只有一个元素为1，其他全部为0，如式子(2.2)所示。类似于深度可分离卷积，对于输入的M个通道的张量，分别对应了M个Shift卷积核，如Fig 2.1的不同颜色的卷积核所示。

$$\tilde{G}_{k,l,m} = \sum_{i,j} \tilde{K}_{i,j,m} F_{k+\hat{i},l+\hat{j},m} \tag{2.1}$$

$$\tilde{K}_{i,j,m} = \begin{cases} 1, & \text{当} i = i_m,\ j = j_m \\ 0, & \text{其他} \end{cases} \tag{2.2}$$

我们把其中一个通道的shift卷积操作拿出来分析，如Fig 2.2所示。我们发现，shift卷积过程相当于将原输入的矩阵在某个方向进行平移，这也是为什么该操作称之为shift的原因。虽然简单的平移操作似乎没有提取到空间信息，但是考虑到我们之前说到的，通道域是空间域信息的层次化扩散。因此通过设置不同方向的shift卷积核，可以将输入张量不同通道进行平移，随后配合1x1卷积实现跨通道的信息融合，即可实现空间域和通道域的信息提取。

Fig 2.2 shift卷积算子中的卷积操作，经过填充后如（2）所示，我们发现，shift卷积相当于将原输入矩阵在某个方向进行平移。

我们发现shift卷积的本质是特定内存的访问，可学习参数只是集中在1x1卷积操作中。因此如果实现得当，shift卷积是不占用额外的计算量和参数量的，结合shift卷积，只使用1x1卷积即可提取到结构化层次化的空间域信息，因此大大减少了卷积网络设计的参数量和计算量。

然而我们注意到，对于一个卷积核大小为D_K，通道数为M的卷积核而言，其可能的搜索空间为$(D_K^2)^M$，在学习过程中穷尽这个搜索空间是不太现实的。为了减少搜索空间，[1]采用了一种简单的启发式设计：将M个通道均匀地分成D_K^2个组，我们将每个组称之为 **平移组**（shift group）。每个组有$\lfloor M/D_K^2 \rfloor$个通道，这些通道都采用相同的平移方向。当然，有可能存在除不尽的情况，这个时候将会有一些通道不能被划分到任意一个组内，这些剩下的通道都称之为“居中”组，如Fig 2.3所示，其中心元素为1，其他为0，也即是对原输入不进行任何处理。

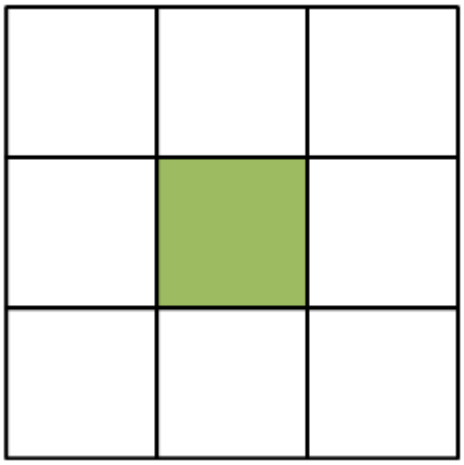

Fig 2.3 居中组的中心元素为1，其他元素都为0。

虽然通过这种手段大大缩小了搜索空间，但是仍然需要让模型学出如何将第m个通道映射到第$n, n \in [0, \lfloor M/D_K^2 \rfloor - 1]$个平移组的最佳排列规则，这仍然是一个很大的搜索空间。为了解决这个问题，以下需要提出一种方法，其能够使得shift卷积层的输出和输入是关于通道排序无关的。假设$\mathcal{K}_\pi(\cdot)$表示是在以π为通道排序

的shift卷积操作，那么公式(2.1)可以表示为$\tilde{G}=\mathcal{K}_{\pi}(F)$，如果我们在进行该卷积之前，先后进行两次通道排序，分别是$\mathcal{P}_{\pi_1}$和$\mathcal{P}_{\pi_2}$，那么我们有：

$$\tilde{G}=\mathcal{P}_{\pi_2}(\mathcal{K}_{\pi}(\mathcal{P}_{\pi_1}(F)))=(\mathcal{P}_{\pi_2}\circ\mathcal{K}_{\pi}\circ\mathcal{P}_{\pi_1})(F) \tag{2.3}$$

其中$\circ$表示算子组合。

Reference

---

[1]. Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., ... & Keutzer, K. (2018). Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9127-9135).

[2]. Cheng, K., Zhang, Y., He, X., Chen, W., Cheng, J., & Lu, H. (2020). Skeleton-Based Action Recognition With Shift Graph Convolutional Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 183-192).

[3]. https://github.com/peterhj/shiftnet_cuda_v2

[4]. Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

[5]. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

[6]. https://baike.baidu.com/item/%E5%B1%80%E9%83%A8%E6%80%A7%E5%8E%9F%E7%90%86

1. 访存指的是从内存中取出操作数加载到寄存器中，通常访存时间远比计算时间长，大概是数量级上的差别。 ↩