KCPStack:张量分解的卷积核分层矩阵压缩方法

王鼎衡[1]，刘保荣[1]，杨维[1]，杨朝旭[2,3]
（1. 西北机电工程研究所，712099，陕西咸阳；2. 西安交通大学航天航空学院，710049，西安；
3. 西安交通大学复杂服役环境重大装备结构强度与寿命全国重点实验室，710049，西安）

摘要：针对现有张量分解卷积核压缩方法难以兼顾时空轻量化、过于依赖卷积瓶颈结构等问题，提出一种具有可观压缩与加速能力的卷积核分层矩阵压缩方法(KCPStack)。首先，在矩阵乘法视角下，将卷积核按通道拆分为2阶克罗内克规范多项式(KCP)分解，所得因子张量组合为两层权重矩阵，使卷积计算转换为具有较高推理效率的双层轻量卷积结构；其次，对比所提KCPStack方法与其他典型张量分解卷积核压缩方法的参数约减空间复杂度与推理计算时间复杂度；最后，基于RK3588神经处理单元进行KCPStack方法的部署，面向实际场景目标检测识别需求开发相关应用。实验结果表明：与现有张量分解方法相比，在张量秩相同或者参数量相当的前提下，所提KCPStack方法具有最快的推理计算效率；在图像分类标准数据集CIFAR-10和ImageNet上，KCPStack方法能够将精度损失控制在1%左右，最高可减少85.0%的参数量和79.8%的计算量；在目标检测识别标准数据集COCO上，KCPStack方法相对于基线模型的平均精度下降不超过1%；采用所提KCPStack方法对实际场景进行目标检测识别，在RK3588神经处理单元上能达到95.4%的平均精度和35帧/s的图像处理帧率，内存开销仅为33.1 MB。

关键词：克罗内克规范多项式张量分解；卷积核压缩；推理效率；分层矩阵；目标检测识别
中图分类号：TP274+.5 **文献标志码**：A
DOI：10.7652/xjtuxb202403013 **文章编号**：0253-987X(2024)03-0137-12

KCPStack: Tensor Decomposed Compression Method with Layered Matrices for Convolutional Kernels

WANG Dingheng[1], LIU Baorong[1], YANG Wei[1], YANG Zhaoxu[2,3]
(1. Northwest Institute of Mechanical & Electrical Engineering, Xianyang, Shaanxi 712099, China;
2. School of Aerospace Engineering, Xi'an Jiaotong University, Xi'an 710049, China; 3. State Key Laboratory for Strength and Vibration of Mechanical Structures, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: To address the limitations of existing tensor decomposition-based methods for compressing convolutional kernels, such as the trade-off between spatial and temporal lightweight and excessive reliance on convolutional bottleneck structure, a compression method with layered matrices called KCPStack is proposed in this paper which offers considerable compression and acceleration capabilities. Firstly, from the perspective of matrix multiplication, the convolutional kernels are split by channel and subjected to a second-order Khatri-Rao Product (KCP) decomposition and the resulting factor tensors are combined into two-layer weight matrices,

收稿日期：2023-08-16。 作者简介：王鼎衡(1988—)，男，助理研究员；杨朝旭(通信作者)，男，副教授，硕士生导师。
基金项目：国家自然科学基金资助项目(12002254)。
网络出版时间：2023-12-05 网络出版地址：https://link.cnki.net/urlid/61.1069.T.20231204.1413.002

thereby transforming the convolutional computation into a two-layer lightweight convolutional structure with higher inference efficiency. Secondly, a comparison is made between the space complexity regarding parameter reduction and time complexity regarding inference computation of the KCPStack method and other typical tensor decomposition-based compression methods for convolutional kernels. Lastly, the KCPStack method is deployed on the RK3588 neural processing unit to develop related applications to meet the object detection and recognition needs for the real scene. The experimental results demonstrate that, compared with existing tensor decomposition-based methods, the proposed KCPStack method achieves the highest inference computation efficiency under the same tensor rank or comparable parameter quantity conditions. On the benchmark datasets CIFAR-10 and ImageNet for image classification, the KCPStack method controls the accuracy loss to around 1% while achieving a maximum parameter reduction of 85.0% and a computational saving of 79.8%. On the benchmark dataset COCO for object detection and recognition, the KCPStack method exhibits an average precision drop of less than 1% compared to the baseline model. When the KCPStack method is adopted for an object detection and recognition task in the real scene, an average precision of 95.4% and of an image processing frame rate of 35 frames per second are achieved on the RK3588 neural processing unit, and it requires only 33.1 MB of memory consumption.

Keywords: KCP tensor decomposition; convolutional kernel compression; inference efficiency; layered matrices; object detection and recognition

卷积神经网络(CNN)在现代计算机视觉与人工智能领域发挥了重要作用,但优越的视觉理解能力往往意味着 CNN 需要可观的参数量与计算量以提供足够的表达能力,这对 CNN 在资源受限条件下的部署提出了挑战[1-3]。为了解决这一问题,学者们提出了大量的 CNN 压缩方法,其中,具有坚实理论基础的张量分解方法[4],凭借其便于实现、压缩比高、可直接训练等优势,在近年受到越来越多的关注[5]。

张量分解用于 CNN 压缩一般有两种方式,分别是张量估计与张量映射[6]。其中,张量估计将矩阵低秩估计的思想拓展至张量层面,将 CNN 中需要压缩的卷积核分解为若干小规模因子张量并作为新权重,使这些因子张量形成对原卷积核的低秩表达[7];张量映射将张量分解得到的因子张量视作多层神经网络模块,通过更多非线性表达能力提升 CNN 压缩效果[8]。张量估计在循环神经网络(RNN)的压缩方面应用较多,可以实现上万倍的压缩比。例如,Yang 等[9]借助张量串(TT)分解实现了上万倍压缩比的轻量化 RNN,其他如 Ye 等[10]、Pan 等[11]、Yin 等[12]分别使用块项(BT)分解、张量环(TR)分解、分层塔克(HT)分解达成了相当甚至更好的轻量化效果。张量估计的轻量化 RNN 近年也产生了一些面向现实业务的应用,如司法案件判定[13]、用户推荐[14]等。张量映射在 CNN 中效果更佳,如 Zhang 等[15]将卷积核以奇异值分解(SVD)方式转换为一个轻量化卷积与 1×1 卷积堆叠的形式,Kim 等[16]将塔克(Tucker)分解映射为 ResNet 瓶颈结构,Chen 等[17]将 BT 分解映射为 ResNeXt 的分组瓶颈结构,Astrid 等[18]和 Lebedev 等[19]将 CP 分解各自映射为 3 层可分离瓶颈结构与 4 层非对称可分离瓶颈结构,文献[20]借助克罗内克规范多项式(KCP)分解提出 4 层非对称变通道可分离瓶颈结构。

上述多种张量估计和张量映射方法各有优劣。例如,张量估计受限于具体张量分解表达能力的差异,在压缩精度与压缩比两方面较难达到均衡;张量映射虽能带来额外非线性表达能力,但往往会将神经网络改造为一个深而窄的新模型,导致训练难度明显提升甚至无法获得好的训练结果。尤其是在面向 CNN 时,现有张量估计方法受卷积特殊的计算方式制约,在推理过程中往往必须先将因子张量恢复为普通卷积核的形式再计算正常卷积;现有张量映射方法大多与已成为常用模块的瓶颈结构相似,故不具备再次压缩 ResNet 等典型 CNN 的能力。虽然文献[21]在 RNN 压缩角度指出 KCP 分解有着兼顾压缩比与推理计算效率的理论潜力,但在实践中受张量转置、多路计算等影响,其实际计算的时

间复杂度仍然很高。总之，现有张量分解卷积核压缩方法难以较好地兼顾时间和空间复杂度，且不便于压缩已经具有瓶颈结构的典型 CNN 模型。

为解决这些问题，本文考虑 KCP 分解可能兼顾压缩比与推理计算效率的潜力，充分研究其因子张量拓扑结构的灵活性，提出一种具有可观压缩与加速能力的卷积核分层矩阵压缩方法 KCPStack；挖掘 KCP 分解数学特性，分析卷积计算与矩阵乘法的关系，提出卷积核以 2 阶 KCP 分解进行压缩并将因子张量转换为双层轻量卷积结构的方法，在理论上分析其推理效率优越性，证明其计算结果合理性；对比 KCPStack 与其他典型张量压缩方法的时空复杂度，并在 CIFAR-10 和 ImageNet 数据集上进行仿真实验；面向实际场景目标检测识别需求，提出 KCPStack 在 RK3588 神经处理单元（NPU）上的部署方法并开发相关应用。

1　KCPStack 卷积核压缩方法

KCPStack 方法旨在将 KCP 压缩的循环神经网络（KCP-RNN）[21] 拓展至卷积核压缩并解决其实际推理计算效率较差的问题。具体思路为：首先将卷积运算按照图像到矩阵（im2col）算法思想转换为矩阵乘法，其次借助 KCP 对卷积核转换后得到的权重矩阵进行 2 阶分解，接着将输入特征数据与 KCP 因子张量之间的运算过程优化为两个连续矩阵乘法，最后通过 im2col 算法的逆变换使卷积核及其计算转变为双层轻量卷积结构。

1.1　KCP 分解

KCP 分解最早由 Phan 等[22] 提出，可将大规模矩阵或高阶张量转换为多组小规模 CP 因子矩阵的形式，从而获得很高的压缩比。具体地，设任意权重矩阵 $\boldsymbol{W}\in\mathbb{R}^{M\times N}$ 的模长可各自分解为 d 个因数之积 $M=\prod_{i=1}^{d}m_i$（一般与输入数据维度相关，可称为输入模）和 $N=\prod_{i=1}^{d}n_i$（一般与输出数据维度相关，可称为输出模），则 $\boldsymbol{W}$ 可重构为 d 阶张量 $\underline{\boldsymbol{W}}\in\mathbb{R}^{m_1n_1\times m_2n_2\times\cdots\times m_dn_d}$ 并分解为

$$\underline{\boldsymbol{W}}=\sum_{k=1}^{K}\underline{\boldsymbol{A}}_k\otimes\underline{\boldsymbol{B}}_k \tag{1}$$

式中：$\otimes$ 表示克罗内克（Kronecker）积；K 为 KT 秩。式（1）的 d 阶张量 $\underline{\boldsymbol{A}}_k\in\mathbb{R}^{m_1\times m_2\times\cdots\times m_d}$ 与 $\underline{\boldsymbol{B}}_k\in\mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ 还存在 CP 分解

$$\begin{cases}\underline{\boldsymbol{A}}_k=\underline{\boldsymbol{E}}^{(A_k)}\cdot\boldsymbol{F}_1^{(A_k)}\cdot\boldsymbol{F}_2^{(A_k)}\cdot\cdots\cdot\boldsymbol{F}_d^{(A_k)}\\ \underline{\boldsymbol{B}}_k=\underline{\boldsymbol{E}}^{(B_k)}\cdot\boldsymbol{F}_1^{(B_k)}\cdot\boldsymbol{F}_2^{(B_k)}\cdot\cdots\cdot\boldsymbol{F}_d^{(B_k)}\end{cases} \tag{2}$$

式中：$\cdot$ 指两个张量之间按长度相同的模进行缩并运算，矩阵乘法即缩并运算的特例；$\boldsymbol{F}_i^{(A_k)}\in\mathbb{R}^{m_i\times R_k^{(A)}}$ 和 $\boldsymbol{F}_i^{(B_k)}\in\mathbb{R}^{n_i\times R_k^{(B)}}$ 是 CP 因子矩阵，可作为权重矩阵分解之后的新权重，$R_k^{(A)}$ 和 $R_k^{(B)}$ 分别是 $\underline{\boldsymbol{A}}_k$ 和 $\underline{\boldsymbol{B}}_k$ 的 CP 秩；$\underline{\boldsymbol{E}}^{(A_k)}$ 与 $\underline{\boldsymbol{E}}^{(B_k)}$ 可视作边长为 $R_k^{(A)}$ 和 $R_k^{(B)}$ 的 d 阶单位超对角张量，在与外部输入计算时可忽略[21]。

KCP 在理论上的高效计算优势在于将输入模和输出模通过 Kronecker 积分隔开，如此一方面可以避免其他张量分解方法（如 TT[9]、BT[10]）的因子张量将输入模与输出模绑定导致计算困难的情况；另一方面提供了足够简洁的因子张量形状和可变的拓扑结构，使得探索输入数据和 KCP 形式权重之间的灵活计算成为可能。

1.2　卷积运算转化为矩阵乘法

任意 S 通道 $H\times W$ 尺寸的特征图可表示为 3 阶张量 $\underline{\boldsymbol{X}}\in\mathbb{R}^{S\times H\times W}$，与高×宽为 $h\times w$ 的 4 阶张量的卷积核 $\underline{\boldsymbol{C}}\in\mathbb{R}^{T\times S\times h\times w}$ 进行卷积计算后可得 T 通道输出特征图 $\underline{\boldsymbol{Y}}\in\mathbb{R}^{T\times H^{\dagger}\times W^{\dagger}}$，即

$$\underline{\boldsymbol{Y}}=\underline{\boldsymbol{X}}*\underline{\boldsymbol{C}} \tag{3}$$

式中：$*$ 表示卷积计算。另外，$H^{\dagger}\times W^{\dagger}$ 表示输出特征图尺寸，其与输入特征图尺寸的关系一般为

$$\begin{cases}W^{\dagger}=((W+2p-(w-1)-1)/s)+1\\ H^{\dagger}=((H+2p-(h-1)-1)/s)+1\end{cases} \tag{4}$$

其中 p 表示输入特征图四周补 0 宽度，s 为卷积扫描步长。

卷积运算在实际中一般转换为矩阵乘法完成，具体是将每个特征图按照对应卷积核扫描位置展开，再将展开后的特征图与卷积核均重构为矩阵，两个矩阵相乘所得结果即为输出特征图的矩阵排列形式，该过程即所谓 im2col 算法[23]，如图 1 所示。卷积转变为矩阵乘法之后，即可使用各类张量分解权

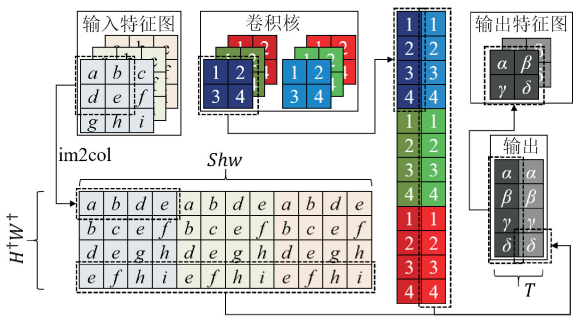


图 1　卷积的 im2col 算法

Fig. 1　The im2col algorithm of convolution

重与输入之间的计算方法，如 TT 序列化缩并[6]、KCP 多路计算[21]、HT 中序遍历[24]等。相比之下，文献[21]研究了 KCP 分解在神经网络压缩方面的潜力，发现其在理论上可同时减少神经网络的参数量和计算量，但其对应的计算方法在实践中并未实现高效性，故本文进一步挖掘 KCP 分解压缩卷积核的潜力和实际加速能力。

1.3 卷积核的 KCP 分层矩阵压缩与推理

卷积计算按照图 1 所示 im2col 转换为矩阵乘法后，参照文献[21]给出的输入特征数据与 KCP 因子之间的可并行算法，如图 2 所示。具体地，令输入特征图按照图 1 的 im2col 算法重构为 $\boldsymbol{X}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times Shw}$，在 2 阶分解 $S=S_1S_2$ 和 $T=T_1T_2$ 约束下，令卷积核 $\boldsymbol{C}$ 重构为 $\boldsymbol{C}^{\dagger}$，即

$$\begin{cases}\rho:\mathbb{R}^{T\times S\times h\times w}\to\mathbb{R}^{Shw\times T}\to\mathbb{R}^{(S_1\cdot S_2hw)\times(T_1\cdot T_2)}\\ \boldsymbol{C}^{\dagger}=\rho(\underline{\boldsymbol{C}})\end{cases}\tag{5}$$

对 $\boldsymbol{C}^{\dagger}$ 作 2 阶 KCP 分解可得 K 组因子 $\boldsymbol{F}_1^{(A_k)}\in\mathbb{R}^{S_1\times R_k^{(A)}}$，$\boldsymbol{F}_1^{(B_k)}\in\mathbb{R}^{T_1\times R_k^{(B)}}$，$\boldsymbol{F}_2^{(A_k)}\in\mathbb{R}^{S_2hw\times R_k^{(A)}}$ 和 $\boldsymbol{F}_2^{(B_k)}\in\mathbb{R}^{T_2\times R_k^{(B)}}$，再令输入 $\boldsymbol{X}$ 的模 Shw 拆开为 $S_1\times S_2hw$ 并依次与各因子张量按照图 2 缩并（图中红色为待缩并模）求和即可得到结果 $\boldsymbol{Y}$。该算法的结果被证明等价于输入和 KCP 恢复的矩阵相乘的结果[21]。若设最大的 S_i 或 T_i 为 D，那么文献[21]算法的时间复杂度为

$$O(H^{\dagger}W^{\dagger}D^2K(hwR^{(A)}+2hwR^{(A)}R^{(B)}+R^{(B)}))\tag{6}$$

然而，图 2 所示算法至少存在两方面的明显缺陷：一是计算过程中同时存在 K 路中间结果 $\underline{\boldsymbol{X}}_k$，这对实际应用中的内存消耗提出巨大挑战；二是图 2 的过程意味着输入及其中间结果需经历多次张量转置以便缩并，这为计算过程带来了无形的额外时间消耗。为了解决这些问题，在此提出一种避免多路中间结果和频繁张量转置的卷积核 KCP 分层矩阵压缩形式，并给出该形式的因子与输入特征图的计算方法。

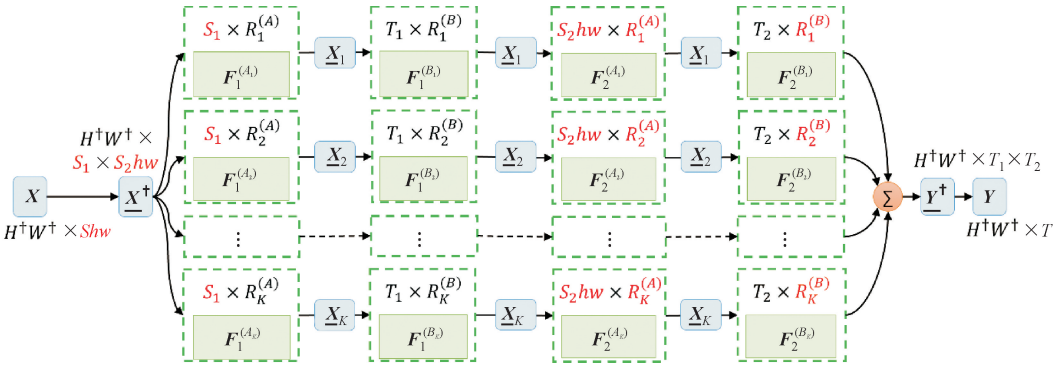


图 2 KCP 卷积核与输入特征图的分路计算[21]

Fig. 2 Computation in separate ways between the convolutional kernel in KCP and the input feature maps[21]

根据文献[21]的公式，图 2 的计算过程可表示为

$$\begin{cases}\underline{\boldsymbol{X}}_k=(((\underline{\boldsymbol{X}}^{\dagger}\cdot\boldsymbol{F}_1^{(A_k)})\otimes\boldsymbol{F}_1^{(B_k)})\cdot(\boldsymbol{F}_2^{(A_k)}))\cdot(\boldsymbol{F}_2^{(B_k)})^{\mathrm{T}}\\ \underline{\boldsymbol{Y}}^{\dagger}=\sum_k\underline{\boldsymbol{X}}_k\end{cases}\tag{7}$$

式中：$\underline{\boldsymbol{X}}^{\dagger}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times S_1\times S_2hw}$、$\underline{\boldsymbol{Y}}^{\dagger}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times T_1\times T_2}$ 分别是特征图矩阵 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 张量化的结果，以便和作为因子的模 S_1、S_2、T_1、T_2 计算。观察式(7)输入输出可知，计算的最终目标是将模 $S_1\times S_2hw$ 转换为 $T_1\times T_2$。因此，利用 KCP 分解松散可变的因子拓扑结构，灵活考虑一种输入与所有因子之间合适的计算顺序来达成同样的目标即可。以此为出发点，式(7)各个因子张量实际可堆叠为分块矩阵如下

$$\begin{cases}\boldsymbol{F}^{(A)}=\begin{bmatrix}\mathrm{vec}(\boldsymbol{F}_1^{(A_1)}\cdot(\boldsymbol{F}_2^{(A_1)})^{\mathrm{T}})\\ \mathrm{vec}(\boldsymbol{F}_1^{(A_2)}\cdot(\boldsymbol{F}_2^{(A_2)})^{\mathrm{T}})\\ \vdots\\ \mathrm{vec}(\boldsymbol{F}_1^{(A_K)}\cdot(\boldsymbol{F}_2^{(A_K)})^{\mathrm{T}})\end{bmatrix}^{\mathrm{T}}\in\mathbb{R}^{Shw\times K}\\ \boldsymbol{F}^{(B)}=\begin{bmatrix}\mathrm{vec}(\boldsymbol{F}_1^{(B_1)}\cdot(\boldsymbol{F}_2^{(B_1)})^{\mathrm{T}})\\ \mathrm{vec}(\boldsymbol{F}_1^{(B_2)}\cdot(\boldsymbol{F}_2^{(B_2)})^{\mathrm{T}})\\ \vdots\\ \mathrm{vec}(\boldsymbol{F}_1^{(B_K)}\cdot(\boldsymbol{F}_2^{(B_K)})^{\mathrm{T}})\end{bmatrix}\in\mathbb{R}^{K\times T}\end{cases}\tag{8}$$

式中：$S=S_1S_2$；$T=T_1T_2$。于是，矩阵化的输入特征图 $\boldsymbol{X}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times Shw}$ 通过简单的分层矩阵运算即可得到矩阵化的输出特征图 $\boldsymbol{Y}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times T}$，即

$$\boldsymbol{Y}=(\boldsymbol{X}\cdot\boldsymbol{F}^{(A)})\cdot\boldsymbol{F}^{(B)} \tag{9}$$

式(9)即是将 KCP 因子重新组合为多层矩阵连乘的形式，本文称之为 KCPStack 方法，如图 3 所示。

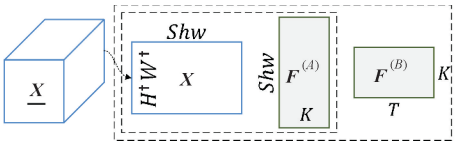


图 3　卷积 KCPStack 方法分层矩阵示意

Fig. 3　Schematic KCPStack method with layered matrices for convolutional kernels

在考虑预先缩并消去 CP 秩的前提下，式(9)的 KCPStack 推理时间复杂度即为

$$O(H^{\dagger}W^{\dagger}D^{2}K(hw+1)) \tag{10}$$

显然，式(10)所示时间复杂度低于文献[21]的式(6)，况且式(9)不涉及张量转置运算，实际计算效率节省会更显著。即使将可预处理的 $\boldsymbol{F}^{(A)}$ 与 $\boldsymbol{F}^{(B)}$ 构建计算也考虑在内，时间复杂度也仅为

$$O(H^{\dagger}W^{\dagger}D^{2}K(hw+1)+D^{2}K(R^{(A)}hw+R^{(B)})) \tag{11}$$

式(11)一方面将特征图尺寸 $H^{\dagger}W^{\dagger}$ 与 CP 秩分隔开，另一方面取消了 $R^{(A)}R^{(B)}$ 的 CP 秩乘积项，故与式(6)相比仍明显优化。需要强调的是，文献[22]指出 KCP 分解还可重写为一种单路的因子稀疏化 CP 从而将式(2)的因子组织为

$$\boldsymbol{F}_i=\left[\boldsymbol{F}_i^{(A_1)}\otimes\boldsymbol{F}_i^{(B_1)}\quad\cdots\quad\boldsymbol{F}_i^{(A_K)}\otimes\boldsymbol{F}_i^{(B_K)}\right] \tag{12}$$

与式(8)相比，式(12)保留了 CP 秩使得计算十分复杂，且合并的因子矩阵中的内容由 Kronecker 积衔接，导致其空间复杂度明显高于式(8)。文献[21]亦证明了式(12)在计算和存储两方面都不占优。

本小节最后给出如下定理证明式(9)的 KCPStack 计算方式的正确性。

定理 1　矩阵化输入特征图 $\boldsymbol{X}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times Shw}$ 重构为张量$\underline{\boldsymbol{X}}^{\dagger}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times S_1\times S_2hw}$ 后，以式(7)的方式进行计算，得到$\underline{\boldsymbol{Y}}^{\dagger}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times T_1\times T_2}$ 再重构为矩阵化输出特征图 $\boldsymbol{Y}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times T}$。$\boldsymbol{X}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times Shw}$ 还可按照式(8)与(9)计算得到同样的输出 $\boldsymbol{Y}\in\mathbb{R}^{H^{\dagger}W^{\dagger}\times T}$。

证明　$$\sum_k\left(\left(\begin{pmatrix}\underline{\boldsymbol{X}}^{\dagger}\cdot\\ \boldsymbol{F}_1^{(A_k)}\end{pmatrix}\otimes\boldsymbol{F}_1^{(B_k)}\right)\cdot(\boldsymbol{F}_2^{(A_k)})\right)\cdot(\boldsymbol{F}_2^{(B_k)})^{\mathrm{T}}=\mathrm{Re}((\boldsymbol{X}\cdot\boldsymbol{F}^{(A)})\cdot\boldsymbol{F}^{(B)}) \tag{13}$$

式中：$\mathrm{Re}(\cdot)$：$\mathbb{R}^{H^{\dagger}W^{\dagger}\times T}\to\mathbb{R}^{H^{\dagger}W^{\dagger}\times T_1\times T_2}$。上式拟在张量元素层面证明其成立，设各个模对应的元素坐标为 $p\in[1,H^{\dagger}W^{\dagger}]$，$s_1\in[1,S_1]$，$s_2\in[1,S_2hw]$，$t_1\in[1,T_1]$，$t_2\in[1,T_2]$，$\gamma_k\in[1,R_k^{(A)}]$，$\tau_k\in[1,R_k^{(B)}]$。

式(13)左侧在元素层面可写为

$$\begin{cases}\underline{Y}^{\dagger}(p,t_1,t_2)=\\ \sum\limits_{k=1}^{K}\sum\limits_{\tau_k=1}^{R_k^{(B)}}\Big(\sum\limits_{s_2=1}^{S_2hw}\sum\limits_{\gamma_k=1}^{R_k^{(A)}}\Big(\Big(\sum\limits_{s_1=1}^{S_1}\underline{X}^{\dagger}(p,s_1,s_2)F_1^{(A_k)}(s_1,\gamma_k)\Big)\\ F_1^{(B_k)}(t_1,\tau_k)\Big)F_2^{(A_k)}(s_2,\gamma_k)\Big)F_2^{(B_k)}(t_2,\tau_k)=\\ \sum\limits_{k=1}^{K}\sum\limits_{\tau_k=1}^{R_k^{(B)}}\sum\limits_{s_2=1}^{S_2hw}\sum\limits_{\gamma_k=1}^{R_k^{(A)}}\sum\limits_{s_1=1}^{S_1}\underline{X}^{\dagger}(p,s_1,s_2)F_1^{(A_k)}(s_1,\gamma_k)\cdot\\ F_1^{(B_k)}(t_1,\tau_k)F_2^{(A_k)}(s_2,\gamma_k)F_2^{(B_k)}(t_2,\tau_k)\end{cases} \tag{14}$$

式(13)右侧中，所有 CP 秩 $R_k^{(A)}$ 和 $R_k^{(B)}$ 均已在式(8)缩并，于是可得

$$\underline{Y}^{\dagger}(p,t_1,t_2)=\mathrm{Re}\Big(\sum_{k=1}^{K}\Big(\sum_{\overline{s_1s_2}=1}^{Shw}X(p,\overline{s_1s_2})F^{(A)}(\overline{s_1s_2},k)\Big)F^{(B)}(k,\overline{t_1t_2})\Big) \tag{15}$$

式中：$\overline{s_1s_2}$ 与$\overline{t_1t_2}$ 指多维坐标[25]，表示两个模拉伸为一个模后的坐标，即$\overline{s_1s_2}=s_1S_2hw+s_2$ 和$\overline{t_1t_2}=t_1T_2+t_2$。根据式(8)的定义，可得

$$\begin{cases}F^{(A)}(\overline{s_1s_2},k)=\sum\limits_{\gamma_k=1}^{R_k^{(A)}}F_1^{(A_k)}(s_1,\gamma_k)F_2^{(A_k)}(s_2,\gamma_k)\\ F^{(B)}(k,\overline{t_1t_2})=\sum\limits_{\tau_k=1}^{R_k^{(B)}}F_1^{(B_k)}(t_1,\tau_k)F_2^{(B_k)}(t_2,\tau_k)\end{cases} \tag{16}$$

式(16)实际上拆开了 $Shw=S_1\cdot S_2hw$ 和 $T=T_1\cdot T_2$，故可取消映射 $\mathrm{Re}(\cdot)$ 的变换，$\sum\limits_{\overline{s_1s_2}=1}^{Shw}$ 亦可拆分为 $\sum\limits_{s_1=1}^{S_1}\sum\limits_{s_2=1}^{S_2hw}$。再考虑到 $X(p,\overline{s_1s_2})=\underline{X}^{\dagger}(p,s_1,s_2)$，将式(16)代入式(15)可得

$$\begin{cases}\underline{Y}^{\dagger}(p,t_1,t_2)=\\ \sum\limits_{k=1}^{K}\Big(\sum\limits_{s_1=1}^{S_1}\sum\limits_{s_2=1}^{S_2hw}\underline{X}^{\dagger}(p,s_1,s_2)\sum\limits_{\gamma_k=1}^{R_k^{(A)}}F_1^{(A_k)}(s_1,\gamma_k)\\ F_2^{(A_k)}(s_2,\gamma_k)\Big)\sum\limits_{\tau_k=1}^{R_k^{(B)}}F_1^{(B_k)}(t_1,\tau_k)F_2^{(B_k)}(t_2,\tau_k)=\\ \sum\limits_{k=1}^{K}\sum\limits_{s_1=1}^{S_1}\sum\limits_{s_2=1}^{S_2hw}\sum\limits_{\gamma_k=1}^{R_k^{(A)}}\sum\limits_{\tau_k=1}^{R_k^{(B)}}\underline{X}^{\dagger}(p,s_1,s_2)F_1^{(A_k)}(s_1,\gamma_k)\cdot\\ F_1^{(B_k)}(t_1,\tau_k)F_2^{(A_k)}(s_2,\gamma_k)F_2^{(B_k)}(t_2,\tau_k)\end{cases} \tag{17}$$

将式(17)与式(14)对比可知式(13)成立,证毕。

1.4 分层矩阵与双层轻量卷积结构

KCPStack 的分层矩阵形式主要源于卷积计算的 im2col 矩阵转化,即式(5)的 $\rho:\mathbb{R}^{T\times S\times h\times w}\rightarrow\mathbb{R}^{Shw\times T}$。观察式(8),将映射 $\rho(\cdot)$ 中的 T 换为 K,则 $\boldsymbol{F}^{(A)}\in\mathbb{R}^{Shw\times K}$,经 $\rho(\cdot)$ 的逆变换可化为 $\underline{\boldsymbol{F}}^{(A)}\in\mathbb{R}^{K\times S\times h\times w}$,即得到一个输入 S 通道、输出 K 通道的 $h\times w$ 卷积核。类似地,$\boldsymbol{F}^{(B)}\in\mathbb{R}^{K\times T}$ 可得输入 K 通道、输出 T 通道的 1×1 卷积核 $\underline{\boldsymbol{F}}^{(B)}\in\mathbb{R}^{T\times K\times1\times1}$。因此,式(9)的 KCPStack 分层矩阵形式实际上可解释为双层轻量卷积架构,即

$$\underline{\boldsymbol{Y}}=(\underline{\boldsymbol{X}}*\underline{\boldsymbol{F}}^{(A)})*\underline{\boldsymbol{F}}^{(B)}\tag{18}$$

需强调,式(18)与式(19)是等价的,计算机底层运算时一般仍需要按 im2col 或类似的方式将特征图展开再执行[23],故式(18)更多的是在概念上解释 KCPStack 的本质,同样也方便利用现有成熟卷积编程模块来实现。图 4 详细描绘了式(3)普通卷积、式(9)分层矩阵和式(18)双层卷积之间的关系,注意 $\underline{\boldsymbol{F}}^{(B)}$ 中的每个 1×1 卷积核并非直接由对应第 k 个 $\mathrm{vec}(\boldsymbol{F}_1^{(A_k)}\cdot(\boldsymbol{F}_2^{(A_k)})^{\mathrm{T}})$ 组成,而是与之呈互相垂直的方向。

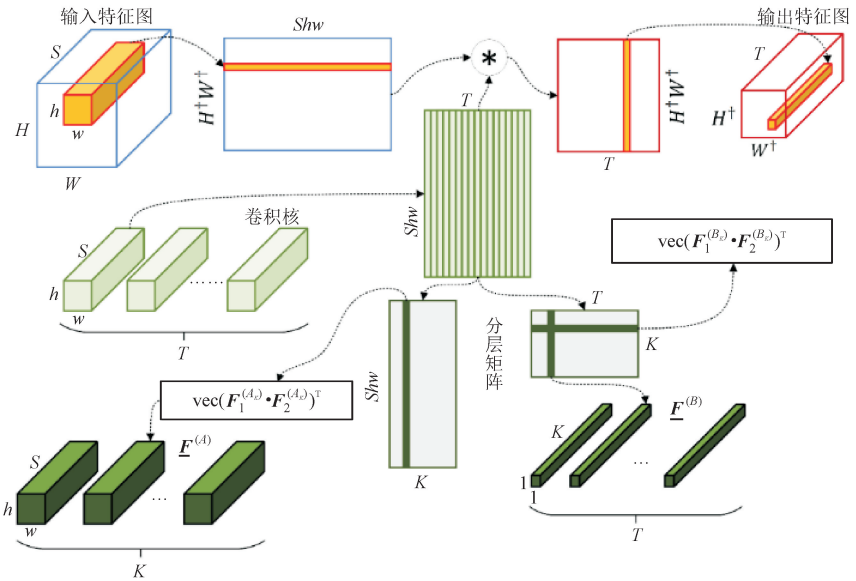


图 4 普通卷积与 KCPStack 双层卷积的变化关系

Fig. 4 The changes in the relationship between common convolution and KCPStack with two-layer convolutions

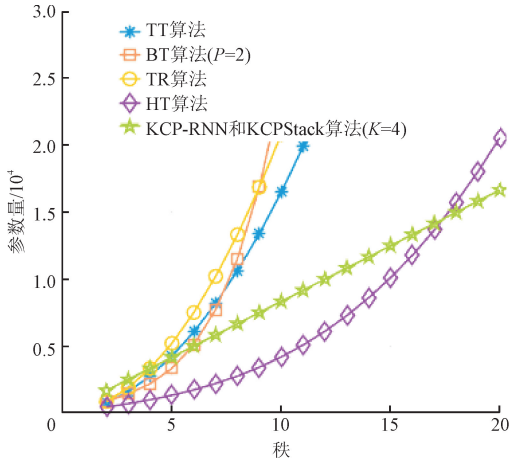
2 KCPStack 方法的压缩效果

本节首先以单个卷积核压缩为仿真对象,分析并验证 KCPStack 方法,并对比其与 TT[9]、BT[10]、TR[11]、HT[12]以及普通 KCP 方法[21]的压缩效果;然后在 CIFAR-10 与 ImageNet 标准数据集上综合考察 KCPStack 方法对 CNN 的压缩效果,并与 SVD[15]、Tucker[16]、BT[17]、CP[18]、克鲁斯卡尔(Kruskal)[19]和 KCPNet[20]等现有典型轻量化 CNN 方法进行对比。

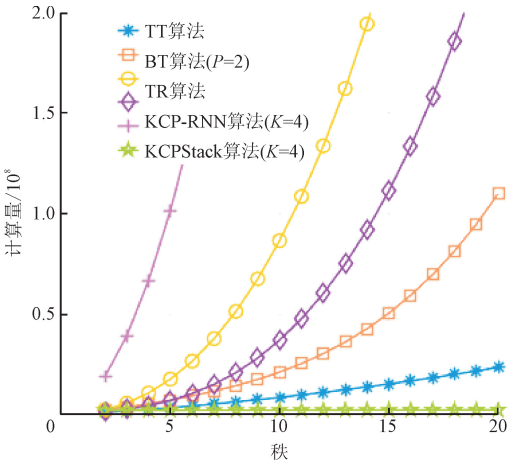
2.1 卷积核压缩与加速

首先考虑 KCPStack 方法对单个卷积核的压缩与加速效果评估,并与其他类型张量分解进行对比。参考工程上常用的大参数量卷积核,设 $h=3$,$w=3$,$C=256$,$S=512$,通过 im2col 算法将其转换为待行 2 阶张量分解的矩阵 $\boldsymbol{C}^{\dagger}\in\mathbb{R}^{(S_1\cdot S_2hw)\times(T_1\cdot T_2)}$,其中,可设 $S_1=16$,$S_2=16$,$T_1=16$,$T_2=32$。注意,KCP 分解按照 2 阶进行,根据式(2)可知,每个 k 将产生 4 个因子,但 TT 等其他张量分解在 2 阶情况下大多只有 2 个因子,为公平对比起见,令其他方法按照 $S_1\times S_2hw\times T_1\times T_2$ 进行 4 阶张量分解。在具体推理算法方面,TT 分解与输入之间考虑文献[7]的序列化缩并算法;BT 分解参照文献[10]取 BT 秩 $B=2$,采用先缩并因子矩阵后缩并高阶核张量的算法;TR 分解使用文献[11]类似于 TT 的序列化缩并;HT 分解采用文献[24]的中序遍历缩并算法;KCP 分解分别使用图 2 的 KCP-RNN 分路计算[21]以及本文所提 KCPStack 分层矩阵计算方式(将构建 $\boldsymbol{F}^{(A)}$ 与 $\boldsymbol{F}^{(B)}$ 的计算也考虑在内),且根据文献[21]取 KT 秩 $K=4$。最后,令不同张量分解的秩(除代表多路求和关系的 BT 秩和 KT 秩)皆从 2 变化至 20 以观察压缩后因子权重的参数量和正向推理的计算量,仿真结果如图 5 所示,其中图 5(a)中所绘 KCP-RNN 与 KCPStack 算

法使用了同一条曲线，这是因为它们都基于 KCP 分解，随秩的变化产生的参数量变化完全一致。



(a)参数量



(a)计算量

图 5　不同张量分解压缩参数量与推理计算量对比

Fig. 5　The comparison of compression parameters and inference computations among different tensor decompositions

图 5 显示，除 HT 分解的参数量在秩较低时表现出了足够的优势之外，KCP 分解(包括 KCPStack 与 KCP-RNN)的参数量随秩的增长呈线性变化，而其他张量分解方法的参数量都表现出一定的指数增长趋势，原因在于 KCP 分解的因子皆为秩一次幂复杂度的矩阵，且能够消解 Tucker 分解无法处理的高阶核张量。在时间复杂度方面，虽然文献[21]认为 KCP-RNN 算法在面向矩阵压缩时能够获得相对最低的计算量，但在压缩卷积核时无法避免式(6)中 hw 与 $R^{(A)}R^{(B)}$ 高度耦合的情况，故其计算量增长反而最为明显。相比之下，本文所提 KCPStack 算法即使在包含构建 $\boldsymbol{F}^{(A)}$ 与 $\boldsymbol{F}^{(B)}$ 计算量的前提下仍有明显优势，若在实际 CNN 的推理中预先定义 $\boldsymbol{F}^{(A)}$ 与 $\boldsymbol{F}^{(B)}$，则推理计算将与 CP 秩无关从而更为显著地节省计算量。值得再次强调的是，图 5 显示所提 KCPStack 方法与文献[21]的 KCP-RNN 具有相同的压缩参数量，这是因为二者采用同样的 KCP 张量分解数据结构；与此对应，本文 KCPStack 方法在推理计算量方面相比 KCP-RNN 有着明显改观。

2.2　CNN 综合压缩效果

为了从 CNN 整体角度更为综合地评判 KCPStack 相对其他张量分解方法的优势，构建 VGG-14 模型，并分别使用现有效果较好的张量映射方法：SVD[15]、Tucker/TT[16](Tucker 分解映射为 3 层卷积瓶颈结构的因子张量等价于 3 阶 TT 分解)、BT[17]、CP[18]、Kruskal[19]、KCPNet[20]以及 KCPStack 方法对其进行压缩。此处 VGG-14 指将通用 VGG-16 模型的多层全连接分类器替换为全局平均池化分类器，如此避免对全连接层的压缩。此外，第一层卷积核因输入通道数过低同样不考虑压缩。为公平对比，通过调整各个方法的秩使得压缩后的模型参数量大体相当；其中，秩的取法主要参考各个方法所引用的文献，本文的 KCPStack 方法则在确保降低计算量的前提下尽量使因子矩阵 $\boldsymbol{F}_1^{(A_k)}$、$\boldsymbol{F}_1^{(B_k)}$、$\boldsymbol{F}_2^{(A_k)}$ 和 $\boldsymbol{F}_2^{(B_k)}$ 保持均衡性，也就是令不同 k 值下的 CP 秩 $R_k^{(A)}$ 与 $R_k^{(B)}$ 相等，同时令 CP 秩与 KT 秩 K 也尽量相等，且 K 不可过低，以使式(18)的双层轻量卷积保持一定的宽度。

另外，针对文献[6]指出张量映射使 CNN 模型加深导致训练难度增大的问题，考虑仿照文献[15-20]，在 KCPStack 引入残差连接，将式(18)改为

$$\underline{\boldsymbol{Y}} = f(g(\underline{\boldsymbol{X}} * \underline{\boldsymbol{F}}^{(A)}) * \underline{\boldsymbol{F}}^{(B)}) + l(\underline{\boldsymbol{X}}) \tag{19}$$

式中：$f(\cdot)$ 为本层原有激活函数；$g(\cdot)$ 为张量映射新添加的线性或非线性激活函数；$l(\cdot)$ 是线性映射 $\mathbb{R}^{S\times H\times W} \to \mathbb{R}^{T\times H^{\dagger}\times W^{\dagger}}$，若本层卷积输出特征图尺寸和通道数不变，该线性映射相当于直接将 $\underline{\boldsymbol{X}}$ 与 $\underline{\boldsymbol{Y}}$ 相加。图 6 描绘了式(19)带残差连接的 KCPStack 结构。

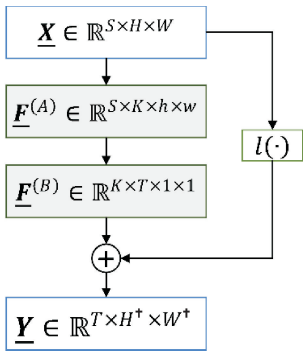


图 6　带残差连接的 KCPStack 结构

Fig. 6　Architecture of KCPStack with residual connection

KCPStack 的卷积核分解详细参数如表 1 所示，特征图边长根据待使用的 CIFAR-10 和 ImageNet 标准图像分类数据集的样本尺寸区别而有所不同。CP 秩记为 R，相关描述方式遵循文献[20]。具体地，$(w\times h,T,z)$ 指卷积窗口尺寸为 $w\times h$，输出通道为 T，卷积扫描步长为 z($z=1$ 则忽略)；多个卷积的组合模块之后的乘数 y 表示该模块重复堆叠 y 次；池化参数将 $(w\times h,T,z)$ 之中的输出通道 T 替换为缩写 MP(最大池化)或 AP(平均池化)，其余意义相同；全连接 $(M\times N)$ 表示其权重矩阵尺寸；→表示特征图边长在此层内发生数值变化。

表 1 KCPStack 压缩的 VGG-14 网络结构

Table 1 Network architecture of VGG-14 compressed by KCPStack

特征图边长		网络模块参数	
CIFAR-10 数据集	ImageNet 数据集	CIFAR-10 数据集	ImageNet 数据集
32	224→122	3×3,64	7×7,64,2
32	122→56		3×3,MP,2
32	56	3×3,64; $K=R=12$	
32→16	56→28	3×3,MP,2	
16	28	(3×3,128)×2; $K=R=16$	
16→8	28→14	3×3,MP,2	
8	14	(3×3,256)×3; $K=R=24$	
8→4	14→7	3×3,MP,2	
4	7	(3×3,512)×3; $K=R=36$	
4→2	7→4	3×3,MP,2	
2	4	(3×3,512)×3; $K=R=44$	
1	1	2×2,AP	4×4,AP
		512×10	512×1 000

分别将 VGG-14 及其各种张量分解压缩的轻量化模型在 CIFAR-10 和 ImageNet 这两个标准图像分类数据集上进行测试，结果如表 2、3 所示。由表 2、3 可知，除无残差连接且存在宽度收紧瓶颈的 SVD 压缩模型之外，其余张量分解方法压缩 VGG 模型的图像分类识别准确率并无明显差异，与基线模型相比的精度损失在 1%左右；在不同方法参数量基本相当的情况下，本文所提 KCPStack 方法的推理计算量除 SVD 外是最低的，在 CIFAR-10 和 ImageNet 上相对基线模型分别减少 85.0%和 82.2%的参数量，分别减少 79.8%和 79.5%的计算量；若与 SVD 一样不考虑残差连接，则 KCPStack 方法的计算量在 CIFAR-10 和 ImageNet 上将分别降至 38.0×10^6 和 159.8×10^6。KCPStack 计算量低的原因在于，和其他张量映射方法相比，式(19)的卷积核 $\underline{\boldsymbol{F}}^{(A)}$ 与 $\underline{\boldsymbol{F}}^{(B)}$ 来自式(8)的 $\boldsymbol{F}^{(A)}$ 与 $\boldsymbol{F}^{(B)}$，后者实际上由 KCP 因子矩阵构成，参数量基于 KCP 因子矩阵而非 $\boldsymbol{F}^{(A)}$ 与 $\boldsymbol{F}^{(B)}$，这使 KT 秩 K 拥有足够的余地来维持

表 2 不同张量分解方法压缩 VGG 模型在 CIFAR-10 数据集上的效果对比

Table 2 The effect comparison of compressing VGG models on CIFAR-10 dataset in different tensor decomposed methods

性能指标	基线模型	SVD[15]	Tucker[16]	BT[17]	CP[18]	Kruskal[19]	KCPNet[20]	KCPStack
准确率/%	93.3±0.05	90.1±0.12	92.7±0.03	93.3±0.01	92.9±0.06	92.9±0.04	93.0±0.01	92.6±0.02
参数量/10^6	14.7	2.2	2.2	2.2	2.2	2.2	2.2	2.2
计算量/10^6	313.2	51.9	78.3	85.0	83.8	83.4	76.6	63.2

表 3 不同张量分解方法压缩 VGG 模型在 ImageNet 数据集上的效果对比

Table 3 The effect comparison of compressing VGG models on ImageNet dataset in different tensor decomposed methods

性能指标	基线模型	SVD[15]	Tucker[16]	BT[17]	CP[18]	Kruskal[19]	KCPNet[20]	KCPStack
准确率/%	68.9±0.02	65.5±0.02	68.0±0.01	68.3±0.01	67.7±0.03	67.4±0.04	67.8±0.03	67.5±0.03
参数量/10^6	15.2	2.7	2.7	2.7	2.7	2.7	2.7	2.7
计算量/10^6	1,232.5	217.6	311.4	337.3	333.0	331.8	298.5	252.6

网络宽度；同时，计算量却是根据式(10)以预先缩并所有 CP 秩的形式统计，即 CP 秩不影响 KCPStack 双层轻量卷积的实际推理效率。简言之，在与其他压缩方法参数量相当的前提下，KCPStack 的 KT 秩 K 确保了足够的神经网络宽度以维持表达能力，同时屏蔽了 CP 秩对推理计算的负担，这是本文所提算法最显著的特点和优势。

3　KCPStack 方法轻量化目标检测识别

本节结合实际场景的具体需求，将 KCPStack 用于已有的 YOLOv5 模型中并在嵌入式设备上部署，实现复杂环境下对特定类别目标的高效检测识别。

3.1　YOLOv5 模型的 KCPStack 压缩方案

YOLOv5 作为较新的单阶段目标检测模型在很多领域得到了应用，本文涉及的实际场景具有对人、车等典型目标的检测识别需求，前期也已在实际场景光电模块中搭载了以国产瑞芯微 RK3588 的 NPU 为部署核心的 YOLOv5s(YOLOv5 系列较轻的变体)目标检测识别模块。然而，在 NPU 上直接运行未压缩的 YOLOv5s 时，目标检测识别的处理帧率难以超过 25 帧/s，不能完全满足实际场景的实时性指标需要。另外，RK3588 对裁剪的支持尚属试用阶段，压缩比和稀疏化程度不够。

为了解决该问题，将 YOLOv5s 中的部分卷积模块以 KCPStack 方法分解，如图 7 所示。需强调，并非所有的卷积模块都需要进行压缩，因为文献[1-2]指出，部分特殊位置的卷积核对模型精度影响较大，如整个 CNN 输入侧的一两层卷积与输出侧的卷积，一般不宜压缩。另外，1×1 卷积作为单纯处理特征图通道变换的特殊模块，如何对其进行压缩仍是需要慎重考虑的开放性问题[1-2]。因此，本文只考虑对 YOLOv5 骨干网络的大部分 3×3 卷积使用 KCPStack 方法进行压缩，即图 7 中的 KCPStack 模块，注意原始 C3 模块中的 1×1 与 3×3 卷积瓶颈可直接替换为 KCPStack 的双层轻量卷积结构，即原有的普通 1×1 与 3×3 卷积堆叠替换为式(8)定义的由分解因子张量构成的 3×3 与 1×1 两层轻量卷积堆叠，C3 模块相应地更名为 KC3 模块。

为更好对比 KCPStack 的性能，与常用的可分离卷积[26]进行对比。可分离卷积与普通卷积通道数相同，但其每个特征图只对应一个卷积扫描窗口，不存在通道之间的交互，故相比普通卷积具有更少的参数量和计算量，常被看作普通卷积的一种轻量化替代。KT 秩 K 的取法考虑为原 3×3 卷积宽度的一半以保证足够的网络宽度，所有因子的 CP 秩取 4、6 或 8 以使参数量小于对应的可分离卷积参数量。

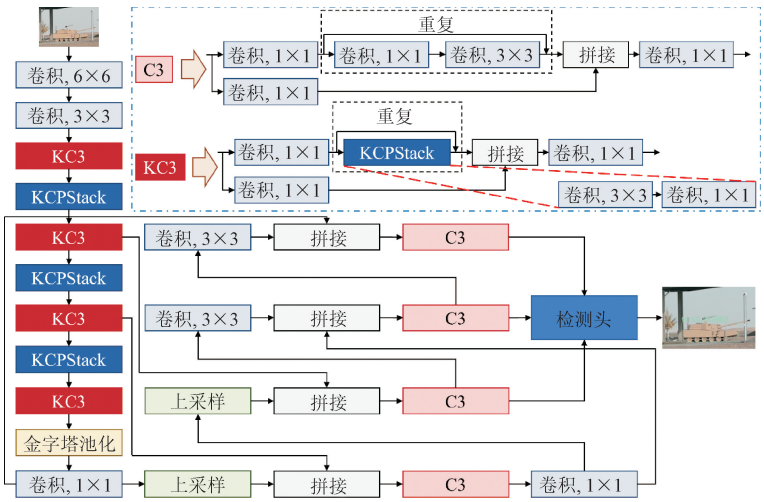


图 7　YOLOv5 的 KCPStack 压缩方案

Fig. 7　The compression scheme with KCPStack for YOLOv5

3.2　KCPStack 方法用于目标检测的效果

记可分离卷积的 YOLOv5s 模型为 DW-YOLOv5s、KCPStack 的 YOLOv5s 模型为 KS-YOLOv5s。基线 YOLOv5s、DW-YOLOv5s 和 KS-YOLOv5s 都先用 COCO 大规模公开数据集预训练，然后使用图 8 所示的复杂环境目标检测专用数据集进行微调训练。从图 8 中可知，专用数据集存在大量远小目标，部分具有与背景相似涂装的目标增大了检测难度。

图 8 复杂环境目标检测识别专用数据集样本

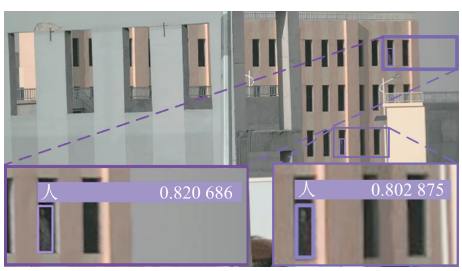
Fig. 8 Samples in the special dataset of object detection and recognition for complex environment

不同目标检测模型在 RK3588 的 NPU 上部署并连续处理 COCO 标准数据集和专用数据集测试图像的综合效果如表 4 所示。由表 4 可见，YOLOv5s、DW-YOLOv5s 和 KS-YOLOv5s 在 COCO 标准数据集上的目标检测识别平均精度逐渐下降，但控制在 1%以内，在专用数据集上的精度损失则收缩至 0.5%以内；作为基线模型的常规 YOLOv5s 具有最高的参数量和内存占用，图像处理速度也最慢；使用可分离卷积压缩的 DW-YOLOv5s 较好地改善了普通 YOLOv5s 的时空复杂度，带来了内存使用和推理效率的提升；而 KS-YOLOv5s 的参数量为最低，内存占用也最小；更重要的是，KS-YOLOv5s 的连续图像处理帧率略超过 35 帧/s，明显优于 DW-YOLOv5s，表明所提方法至少在该任务中比目前常用的可分离卷积具有更佳的时空轻量化能力，更好地满足了实际场景目标检测识别的实时性要求。KS-YOLOv5s 对远小目标检测效果的可视化展示图 9 所示，由图可见，KS-YOLOv5s 能够对隐蔽、杂物遮挡、远距离及低能见度条件下的各类目标实现很好的检测识别效果，且识别置信度大多高于 80%。

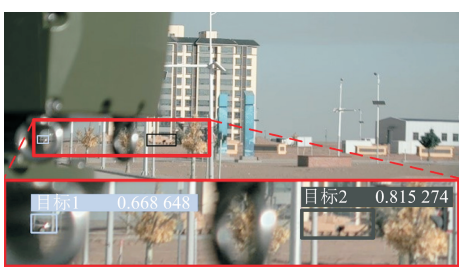
表 4 YOLOv5s 及其轻量化模型在 RK3588 上的 COCO 数据集和专用数据集目标检测识别对比结果

Table 4 The comparison of object detection and recognition results between YOLOv5s and lightweight models on RK3588 for COCO dataset and the special dataset

模型	COCO 数据集平均精度/%	专用数据集平均精度/%	参数量/10^6	内存/MB	单帧平均用时/ms	处理帧率/(帧·s^{-1})
YOLOv5s	56.8±0.01	95.9±0.02	211.9	40.8	41	>24
DW-YOLOv5s	56.2±0.01	95.6±0.03	176.7	34.5	32	>31
KS-YOLOv5s	55.8±0.02	95.4±0.02	164.3	33.1	28	>35

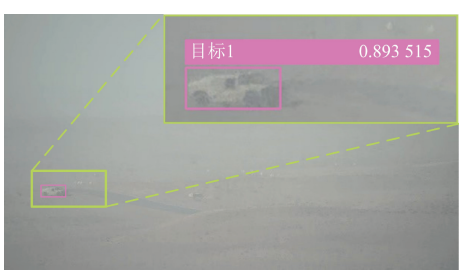


(a)识别人的置信度分别为 82.07%和 80.29%



(b)识别目标 1 和目标 2 的置信度分别为 66.86%和 81.53%

(c)识别目标 1 和目标 2(4 个)的置信度分别为 80.01%和 80.29%、83.66%、81.63%、62.95%



(d)识别目标 1 的置信度为 89.35%

图 9 KS-YOLOv5s 在专用数据集上的目标检测识别效果

Fig. 9 The effect of object detection and recognition with KS-YOLOv5s on the special dataset

综合而言：针对实际的特定场景目标检测任务，KCPStack 方法构建的 KS-YOLOv5s 模型与常规 YOLOv5s 模型相比，仅牺牲 0.5%左右平均精度，却换来约 10 帧/s 的处理速度提升；KS-YOLOv5s 模型与可分离卷积压缩的 DW-YOLOv5s 相比，以 0.2%左右的精度差实现了超过 35 帧/s 的图像处理帧率；所提 KS-YOLOv5s 既保持了对人、车等典型目标的识别精度，又更好地满足了图像处理实时性的要求。

但需要强调的是，YOLOv5s 的轻量化并不像表 2、3 的 VGG 那样可以显著节省模型参数。原因在于，未压缩的 1×1 卷积核实际上贡献了 YOLOv5s 基线模型的绝大部分参数量，而 VGG 模型中并未使用大量的 1×1 卷积，该现象在文献[20]也有提及。然而，如前所述，压缩 1×1 卷积仍然是有挑战性的问题[1-2]。根据作者对国内外研究现状调研所知，目前很少有工作考虑研究压缩 1×1 卷积对整个 CNN 产生的影响，因此，出于面向实际应用的谨慎态度，本文暂不考虑对 1×1 卷积进行压缩。

4　结　论

本文针对卷积核轻量化引入 KCP 张量分解，提出同时具有压缩与加速能力的 KCPStack 卷积核分层矩阵压缩方法，并理论证明其加速算法的成立性。实验表明，所提方法随着秩的增长能够保持高效的参数量与计算量节省能力；面向 CIFAR-10 和 ImageNet 标准数据集压缩 VGG 模型分别可减少 85.0%和 82.2%的参数量，减少 79.8%和 79.5%的计算量；在 RK3588 上能够令 YOLOv5s 的目标检测识别推理帧率超过 35 帧/s，面向 COCO 标准数据集和实际场景专用数据集时分别保持了55.8%和 95.4%的平均精度，且对实际的特定场景下的隐蔽、遮挡、低能见度等远小目标实现了较好的检测识别效果。

KCPStack 方法展现了 KCP 张量分解灵活的拓扑结构变化特点及其轻量化神经网络模型推理计算效率优化的能力，如何将这些优势拓展至 1×1 卷积、Transformer 自注意力等模块，是值得进一步研究的方向。

参考文献：

[1] DENG Lei, LI Guoqi, HAN Song, et al. Model compression and hardware acceleration for neural networks: a comprehensive survey [J]. Proceedings of the IEEE, 2020, 108(4): 485-532.

[2] WU Yang, WANG Dingheng, LU Xiaotong, et al. Efficient visual recognition: a survey on recent advances and brain-inspired methodologies [J]. Machine Intelligence Research, 2022, 19(5): 366-411.

[3] 郭朝鹏，王馨昕，仲昭晋，等. 能耗优化的神经网络轻量化方法研究进展 [J]. 计算机学报，2023，46(1)：85-102.
GUO Chaopeng, WANG Xinxin, ZHONG Zhaojin, et al. Research advance on neural network lightweight for energy optimization [J]. Chinese Journal of Computers, 2023, 46(1): 85-102.

[4] 林景栋，吴欣怡，柴毅，等. 卷积神经网络结构优化综述 [J]. 自动化学报，2020，46(1)：24-37.
LIN Jingdong, WU Xinyi, CHAI Yi, et al. Structure optimization of convolutional neural networks: a survey [J]. Acta Automatica Sinica, 2020, 46(1): 24-37.

[5] WANG Maolin, PAN Yu, XU Zenglin, et al. Tensor networks meet neural networks: a survey and future perspectives [EB/OL]. (2023-05-08) [2023-07-01]. https://arxiv.org/abs/2302.09019.

[6] WANG Dingheng, ZHAO Guangshe, CHEN Hengnu, et al. Nonlinear tensor train format for deep neural network compression [J]. Neural Networks, 2021, 144: 320-333.

[7] NOVIKOV A, PODOPRIKHIN D, OSOKIN A, et al. Tensorizing neural networks [C]//Proceedings of the 28th International Conference on Neural Information Processing Systems. Cambridge, MA, USA: MIT Press, 2015: 442-450.

[8] KOSSAIFI J, TOISOUL A, BULAT A, et al. Factorized higher-order CNNs with an application to spatio-temporal emotion estimation [C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ, USA: IEEE, 2020: 6059-6068.

[9] YANG Yinchong, KROMPASS D, TRESP V. Tensor-train recurrent neural networks for video classification [C]//Proceedings of the 34th International Conference on Machine Learning. Piscataway, NJ, USA: IEEE, 2017: 3891-3900.

[10] YE Jinmian, WANG Linnan, LI Guangxi, et al. Learning compact recurrent neural networks with block-term tensor decomposition [C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ, USA: IEEE, 2018: 9378-9387.

[11] PAN Yu, XU Jing, WANG Maolin, et al. Compressing recurrent neural networks with tensor ring for action recognition [C]//Proceedings of the AAAI Con-

ference on Artificial Intelligence. Palo Alto, CA, USA: AAAI Press, 2019: 4683-4690.

[12] YIN Miao, LIAO Siyu, LIU Xiaoyang, et al. Towards extremely compact RNNs for video recognition with fully decomposed hierarchical tucker structure [C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ, USA: IEEE, 2021: 12080-12089.

[13] 李大鹏，陈剑，王晨，等. 基于模板张量分解和双向 LS TM 的司法案件罪名认定 [J]. 电子学报，2021，49(4)：760-767.
LI Dapeng, CHEN Jian, WANG Chen, et al. Conviction in judicial cases based on template tensor decomposition and bidirectional LSTM [J]. Acta Electronica Sinica, 2021, 49(4): 760-767.

[14] 李晶晶，夏鸿斌，刘渊. 融合注意力 LSTM 的神经张量分解推荐模型 [J]. 中文信息学报，2021，35(5)：91-100.
LI Jingjing, XIA Hongbin, LIU Yuan. Neural tensor factorization recommendation model based on attention LSTM [J]. Journal of Chinese Information Processing, 2021, 35(5): 91-100.

[15] ZHANG Xiangyu, ZOU Jianhua, MING Xiang, et al. Efficient and accurate approximations of nonlinear convolutional networks [C]//2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ, USA: IEEE, 2015: 1984-1992.

[16] KIM Y D, PARK E, YOO S, et al. Compression of deep convolutional neural networks for fast and low power mobile applications [C]//International Conference on Learning Representations (ICLR). San Juan, Puerto Rico: Open Review, 2016: 1-16.

[17] CHEN Yunpeng, JIN Xiaojie, KANG Bingyi, et al. Sharing residual units through collective tensor factorization to improve deep neural networks [C]//Proceedings of the 27th International Joint Conference on Artificial Intelligence. Palo Alto, CA, USA: AAAI Press, 2018: 635-641.

[18] ASTRID M, LEE S I. CP-decomposition with tensor power method for convolutional neural networks compression [C]//2017 IEEE International Conference on Big Data and Smart Computing (BigComp). Piscataway, NJ, USA: IEEE, 2017: 115-118.

[19] LEBEDEV V, GANIN Y, RAKHUBA M, et al. Speeding-up convolutional neural networks using finetuned CP-decomposition [C]//International Conference on Learning Representations (ICLR). San Diego, CA, USA: Open Review, 2015: 1-11.

[20] 王鼎衡，赵广社，姚满，等. KCPNet:张量分解的轻量卷积模块设计、部署与应用 [J]. 西安交通大学学报，2022，56(3)：135-146.
WANG Dingheng, ZHAO Guangshe, YAO Man, et al. KCPNet: design, deployment, and application of tensor-decomposed lightweight convolutional module [J]. Journal of Xi'an Jiaotong University, 2022, 56(3): 135-146.

[21] WANG Dingheng, WU Bijiao, ZHAO Guangshe, et al. Kronecker CP decomposition with fast multiplication for compressing RNNs [J]. IEEE Transactions on Neural Networks and Learning Systems, 2023, 34(5): 2205-2219.

[22] PHAN A H, CICHOCKI A, TICHAVSK P, et al. From basis components to complex structural patterns [C]//2013 IEEE International Conference on Acoustics, Speech and Signal Processing. Piscataway, NJ, USA: IEEE, 2013: 3228-3232.

[23] CHETLUR S, WOOLLEY C, VANDERMERSCH P, et al. cuDNN: efficient primitives for deep learning [EB/OL]. (2014-12-18)[2023-07-01]. https://arxiv.org/abs/1410.0759.

[24] WU Bijiao, WANG Dingheng, ZHAO Guangshe, et al. Hybrid tensor decomposition in neural network compression [J]. Neural Networks, 2020, 132: 309-320.

[25] DOLGOV S V, SAVOSTYANOV D V. Alternating minimal energy methods for linear systems in higher dimensions [J]. SIAM Journal on Scientific Computing, 2014, 36(5): A2248-A2271.

[26] SANDLER M, HOWARD A, ZHU Menglong, et al. MobileNetV2: inverted residuals and linear bottlenecks [C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ, USA: IEEE, 2018: 4510-4520.

（编辑 亢列梅）