计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(2): 532−541, 2025

KCNN: 一种神经网络轻量化方法和硬件实现架构

陈桂林　王观武　王　康　胡敏慧　邓俊荃
（国防科技大学第六十三研究所　南京　210007）
（cglnudt@163.com）

KCNN: A Neural Network Lightweight Method and Hardware Implementation Architecture

Chen Guilin, Wang Guanwu, Wang Kang, Hu Minhui, and Deng Junquan
（*The 63rd Research Institute*, *National University of Defense Technology*, *Nanjing* 210007）

Abstract　Convolutional neural network (CNN) has become one of the most important machine learning technologies in the field of image recognition. In recent years, with the increasing demand for CNN deployment at the mobile edge, the lightweight of CNN has become a research hotspot. The mainstream CNN lightweight methods include pruning and quantization, both of which can effectively reduce the calculation and storage overhead for CNN inference. However, none of these methods fully exploits the bilateral sparsity (weight sparsity and activation sparsity) and potential data-reuse in CNN. Therefore, in order to solve these problems, we propose a new neural network lightweight method, and the *k*-means algorithm is used to cluster the non-zero values of convolution kernel and feature map, and the CNN inference only uses limited cluster values as multipliers to complete all convolutional calculations. Compared with the computational complexity of original convolutional layer $O(n^3)$, the computational complexity of the convolutional layer after lightweight is $O(n^2)$, effectively reducing the amount of computation. Similarly, the non-zero weights of the fully connected layer are also clustered, and only the cluster values and corresponding index vectors are stored on chip, which significantly reduces the storage overhead. Finally, a customized architecture KCNN is designed for this lightweight method. The architecture modularizes the different processes of the CNN, and compared with the previous accelerator, a non-zero clustering module is added. In addition, some caches are added to make use of the data-reuse in the CNN after clustering. The experiment results show that without losing the derivation accuracy, the overall calculation of AlexNet network is reduced by 66%, and the storage expense is reduced by 85%.

Key words　*k*-means; CNN; lightweight; acceleration; compression

摘　要　卷积神经网络（convolutional neural network, CNN）已成为图像识别领域最重要的一项机器学习技术. 近年来, 随着 CNN 在边缘端部署的需求越来越多, CNN 的轻量化也成为研究热点. 主流的 CNN 轻量化方法包括剪枝和量化, 这 2 项技术都能有效地减少 CNN 推导过程中计算和存储开销. 然而, 这些方法未能完全挖掘 CNN 中的双边稀疏性（权重稀疏和激活值稀疏）和潜在的数据复用. 因此, 为了解决上述问题, 提出一种全新的神经网络轻量化方法, 通过 *k*-means 算法对卷积核和特征图的非 0 值进行聚类, 整个神经网络的推导过程中只使用有限的聚类值作为乘数去完成全部卷积计算. 与以往卷积层计算复杂度 $O(n^3)$ 相比, 轻量化处理后的卷积层计算复杂度仅为 $O(n^2)$, 大幅度减少了计算量. 同时, 将全连接层权重也进行非 0 值聚类处理, 片上只存储聚类值和对应的索引向量, 极大地减少存储开销. 最后, 针对该轻量

收稿日期: 2023-05-26; **修回日期**: 2024-01-04
基金项目: 国家自然科学基金项目（61901497）
This work was supported by the National Natural Science Foundation of China (61901497).
通信作者: 王观武（wangguanwu17@nudt.edu.cn）

化方法设计一种硬件实现架构 KCNN. 该架构将 CNN 中的不同处理流程模块化实现，与以往的实现架构相比增加一个非 0 值聚类模块，此外还设计了一些缓存来利用聚类后 CNN 中的数据复用. 实验结果表明在不损失推导精度的情况下，AlexNet 网络整体计算量减少 66%，存储开销减少 85%.

关键词 *k*-means；卷积神经网络；轻量化；加速；压缩

中图法分类号 TP183

DOI: 10.7544/issn1000-1239.202330409 **CSTR:** 32373.14.issn1000-1239.202330409

深度神经网络作为机器学习的典型代表已经在多个应用领域取得巨大成功，例如图像识别[1]、语音识别[2]、自然语言处理[3]等. 其中卷积神经网络大量应用于图像识别领域，典型的卷积神经网络（convolutional neural network, CNN）如图 1 所示，主要由多个卷积层和全连接层组成，其中卷积层具有计算密集性的特点，全连接层具有存储密集性的特点. 近年来，随着神经网络越来越多地部署在边缘侧和移动端设备中，对其轻量化的要求也越来越高. 主流的神经网络轻量化方法包括剪枝和量化[4-8]. 研究表明适当的剪枝不会影响神经网络的性能，并且还有防止网络过拟合的作用[9]，剪枝后的神经网络是一个稀疏神经网络，通过设计特定的运算处理单元可以减少神经网络推导过程中的乘法操作[10-12]. 量化主要通过降低操作数的精度实现. 已有研究表明[13]，适当地降低神经网络操作数精度不会影响整个网络的推导精度，表 1 显示了 2 种不同精度操作数在 MNIST 数据集上的精度对比.

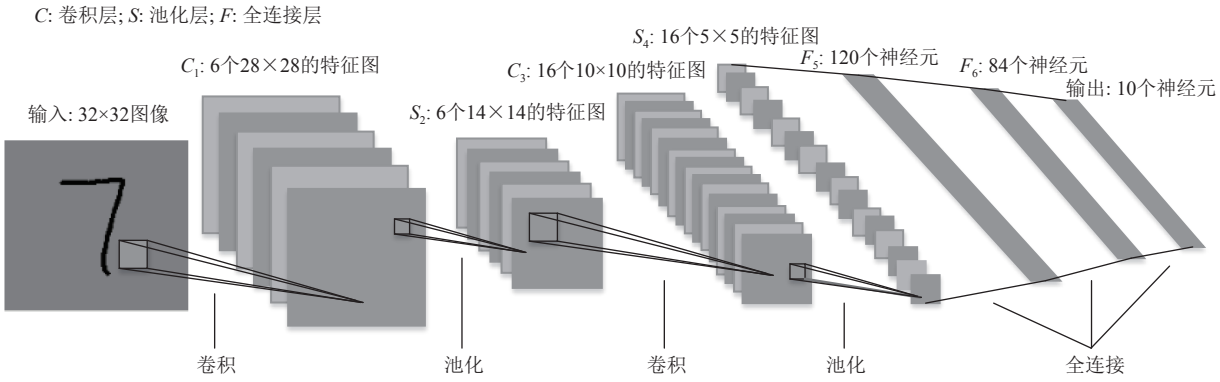


Fig. 1 A representative CNN structure—LeNet-5

图 1 一个具有代表性的 CNN 结构——LeNet-5

Table 1 Effect of Operand Precision on Recognition Accuracy

表 1 操作数精度对识别准确率的影响

精度	错误率
32 b 浮点	0.031 1
16 b 定点	0.033 7

除了上述的剪枝和量化之外，根据神经网络的运算特性（采用 ReLU 作为激活函数的神经网络具有大量 0 值、卷积过程中的数据复用），还可以进一步利用特征图的稀疏性[14]和卷积过程中的数据复用以实现网络的轻量化. 以往许多研究已经通过设计不同的数据流[15-20]分别实现了卷积过程中的权重复用和输入激活值复用. 其中杜克大学的 Edward 等人[21]在 ISCA2022 会议中提出了一种级联结构化的剪枝（cascading structured pruning, CSP）方法，可以保留稀疏神经网络中更多的数据复用机会. 但是这些轻量化方法都没能完全挖掘卷积神经网络中的双边稀疏性（即权重的稀疏和特征图上激活值的稀疏）和潜在的数据复用，目前也没有同时利用双边稀疏性和所有数据复用的 CNN 加速器[22].

通过对已有硬件加速神经网络方法的研究[23]，为解决目前已有轻量化算法和硬件加速器的局限，最大化挖掘神经网络的稀疏特性和数据复用特性，本文提出了一种新的轻量化方法和相应的硬件实现架构 KCNN. 具体做法为对输入特征图在卷积计算前进行非 0 值聚类预处理，然后将这些输入特征图的非 0 聚类值和经过同样聚类处理的卷积核相乘，其得到的结果即为卷积计算所有的乘法计算结果，将对应的乘法结果累加激活即得到了输出特征图. 同理，将

全连接的权重也进行非 0 值聚类和压缩编码, 加速器只存储聚类值和对应索引向量. 这样做的优点是:

1) 减少了计算量. 通过对聚类值和乘法结果地复用, 卷积的计算复杂度由 $O(n^3)$变为 $O(n^2)$, 如果对卷积核也进行聚类处理, 卷积层的计算复杂度可以简化为 $O(n)$.

2) 减少了卷积实现时的数据移动. 有限的乘法结果避免了权重或输入激活值在卷积计算过程中的频繁移动.

3) 减少了存储开销. 聚类后的数据只需要存储索引向量和聚类值, 存储开销大幅度减少.

最后, 我们在多个流行网络(LeNet-5, VGG, AlexNet)和数据集(MNIST, CIFAR-10)中部署了所提出的轻量化算法, 在保持网络推导精度不变的情况下, AlexNet 在处理 CIFAR-10 数据集时计算量减少了 66%, 存储开销减少了 85%.

1 相关工作

CNN 是一个计算密集型和存储密集型人工智能方法, 其中 90% 左右的计算集中在卷积层, 90% 左右的存储集中在全连接层, CNN 的层数越深, 卷积计算量的比例越大, 因此减少卷积层的计算和全连接层的存储是神经网络轻量化的重要研究方向[24]. 目前减少卷积层计算的主流方法包括消除 0 值计算和数据复用[4-8], 减少存储的常用方法是网络稀疏化和数据量化[10-12]. 但是这些做法除了消除部分的 0 值计算外, 其余的计算量并没有改变. 此外, 已有的神经网络稀疏化方法没有完全挖掘 CNN 中的双边稀疏性, 已有的数据量化方法也无法显著地降低存储开销. 例如谷歌 TPU[25] 支持 8 b 低精度运算, 但是其片上存储高达 24 MB, 存储面积占据整个芯片面积的 1/3.

表 2 列出了 3 个典型 CNN 的卷积层和全连接层的计算量, 在过去的图像识别比赛中, 往往网络的深度越大, 识别的准确率越高, 这就导致了网络的参数越来越多, 计算量越来越大. 本文提出一种新的轻量化方法, 通过挖掘更深层次的 CNN 稀疏特性和数据复用特性显著地降低 CNN 的计算开销和存储开销.

Table 2 Weights Data Size and Floating-point Operands of Common Networks

表 2 常见网络的权重数据大小和浮点操作数

CNN	权重数据大小	浮点计算次数	卷积层权重比重/%	全连接层权重比重/%	卷积层计算比重/%	全连接层计算比重/%
LeNet-5	240 KB	2.2 MFLOPs	4.17	95.83	82.4	17.6
AlexNet	240 MB	832 MFLOPs	6.01	93.99	93.2	6.8
VGG16	528 MB	14.4 GFLOPs	10.64	89.36	99.3	0.7

在介绍轻量化算法之前, 我们先回顾一下神经网络的部分特性[26]. 众所周知, 神经网络由于训练方式和测试方式的不同(比如学习率的选择、梯度函数的选择等), 每次得到的网络识别准确率并不是一个特定的值, 而是一个精度范围, 我们将这个精度范围叫做神经网络的置信区间. 神经网络经过轻量化处理后, 如果精度仍然处于置信区间内, 则认定该轻量化方法不影响神经网络的推导精度, 为有效方法. 表 3 列出了部分神经网络在指定数据集上的置信区间. 本文的轻量化方法正是基于神经网络置信区间特性的前提下提出的.

Table 3 Confidence Interval for a Partial Neural Network

表 3 部分神经网络的置信区间 %

CNN(数据集)	识别准确率	置信区间
LeNet-5(MNIST)	99.20	0.21
AlexNet(CIFAR-10)	82.33	0.90
VGG16(CIFAR-10)	91.30	0.80
AlexNet(ImageNet)	58.00(Top 1) 86.86(Top 5)	0.81

2 利用 *k*-means 算法来实现稀疏神经网络的轻量化

在本节中, 我们介绍一种新的神经网络轻量化方法, 通过结合以前在图像对抗中的研究成果[27], 提出将无监督学习中的聚类算法运用到神经网络中, 通过对非 0 聚类值的数据复用, 达到神经网络轻量化的目的.

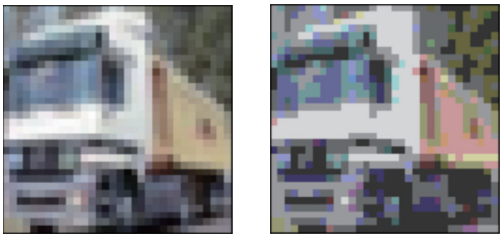
2.1 轻量化原理

首先, 在图像对抗领域, 通过对图像微调, 比如给图像样本增加一些扰动, 会导致神经网络将图像错误分类[28], 比如将猫的图片错误地识别为狗, 将坦克识别为正常汽车等. 我们通过研究发现, 这是由于神经网络的拟合特性决定的, 过多不重要的细节造成了神经网络的错误识别. 这是因为 CNN 的本质是提取图像特征, 而后通过多种特征串联进行图像的判别. 我们通过一种聚类方法, 将图像的某些不重要特征模糊化, 只保留重要特征, 这样通过对识别图像进行简单地预处理, 就可以将微调后猫的照片仍然正确地识别为猫.

常用的聚类算法可以分为基于划分的聚类方法、基于密度的聚类方法、基于层次的聚类方法、基于

联系的聚类方法、基于模型的聚类方法，详细的聚类方法对比见文献 [29]. 在神经网络中，对权重的聚类是一个典型的基于划分聚类. 与其他聚类方法相比，基于划分的聚类方法算法复杂度低、原理简单，本文选择划分聚类中最典型的 *k*-means 聚类算法[30] 作为图像的预处理方法. *k*-means 算法的原理：首先从 n 个数据对象中任意选择 k 个对象作为初始聚类中心，通过迭代 t 次，将类的中心点移动到所包含成员的平均位置，然后重新划分其内部成员，直到所有成员到所属类的距离最近为止. 该算法的算法复杂度为 $O(nkt)$，由于 k 和 t 是可确定的常数，因此算法复杂度可看作是线性的 $O(n)$. *k*-means 算法还有一种优化方法，当需要聚类的数据量太大时，聚类时间会很长. 因此，为了优化算法以减少执行时间，我们从整体中随机抽样，选取部分数据来代替整体. 对于每一个抽样小批量，通过计算平均值得到更新质心，并把小批量里的数据分配给该质心，随着迭代次数的增加，这些质心的变化逐渐减小，直到质心稳定或达到指定的迭代次数时停止计算. 这种优化算法叫做 MiniBatch-KMeans.

该项技术可应用到神经网络的轻量化中，我们在之前已做过类似研究[31]，将整个卷积的输入特征图进行聚类，并将相应聚类值和卷积核相乘得到输出特征图，但是这样会导致原先为 0 的激活值聚类后不为 0，整个特征图由稀疏矩阵变为非稀疏矩阵. 因此，本文提出一种新的改进方法，通过对卷积层输入特征图进行非 0 值聚类，将原先大量的输入数据聚类成 k 类数据，这 k 类数据值和卷积核相乘的结果可以代替原先整个输入特征图和卷积核相乘的结果. 同样地，经过实验证明（详见第 4 节）该方法也可以用到训练好的网络权重数据中. 图 2 是在 CIFAR-10 数据集[32] 中做的简单实验，图 2(a) 是卡车原图，图 2(b) 是将图像聚成 4 类值以后得到的新图像，训练好的神经网络 LeNet-5 在识别时仍可将其正确地识别为卡车.



(a) 卡车原图　　(b) 聚类后的卡车图

Fig. 2　Comparison diagram of a truck image before and after clustering in CIFAR-10 dataset

图 2　CIFAR-10 数据集中 1 张卡车图片在聚类前后的对比图

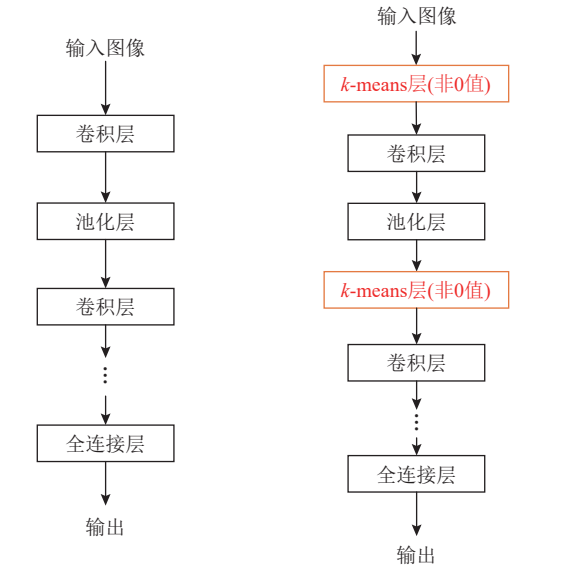
2.2　轻量化加速

由于 CNN 的计算大量集中在卷积层，因此主要针对卷积层计算做轻量化加速. 卷积层的计算公式如式 (1) 所示，$f()$ 是非线性激活函数，β^{f_i} 表示偏移值. 其中 $out(x,y)$ 表示在输出特征图坐标 (x,y) 处的值，$w(k_x,k_y)$ 表示卷积核坐标 (k_x,k_y) 上的权重值，$in(x+k_x, y+k_y)$ 表示输入特征图坐标 $(x+k_x,y+k_y)$ 上的输入值. K_x,K_y 表示卷积核的大小，f_i 表示第 i 幅输入特征图，N_{f_i} 表示输入特征图的数目.

$$out(x,y)=f\left(\sum_{f_i=0}^{N_{f_i}}\left(\sum_{k_x=0}^{K_x}\sum_{k_y=0}^{K_y}w_{f_i}(k_x,k_y)\times in(x+k_x,y+k_y)^{f_i}+\beta^{f_i}\right)\right). \tag{1}$$

卷积层计算可以看作是 2 个矩阵相乘，计算复杂度为 $O(n^3)$，我们通过对输入特征图的聚类，将卷积层计算简化为向量和矩阵相乘，算法复杂度降为 $O(n^2)$. 如果再将卷积核也进行聚类操作，那么最终的卷积层运算将简化为向量和向量相乘，算法复杂度为 $O(n)$.

一个典型的 CNN 流程如图 3(a) 所示，我们在卷积层的前面插入自定义的 *k*-means 聚类层，如图 3(b) 所示. *k*-means 聚类层不需要进行权重训练，它的作用是将上一层的输出特征图进行非 0 值聚类，然后将其作为下一卷积层的输入特征图. 这样处理后 CNN 在做卷积计算时先将 k 个聚类值和卷积核进行乘法计算，将得到的计算结果编号存储，在实际进行特征图卷积处理时将对应的计算结果直接读取使用即可.



(a) 典型CNN处理流程　　(b) 添加*k*-means层的CNN处理流程

Fig. 3　Processing flow of CNN

图 3　CNN 的处理流程

具体操作流程如图 4 所示, 一个3×3×3的卷积核以步长为 1 去卷积一个四周都填充 0 值的32×32×3输入特征图, 最终得到一个32×32的输出特征图, 轻量化处理前该卷积过程中乘法操作的数量为32×32×3×3×3共 27 648 次. 轻量化处理后的乘法操作为4×3×3×3共 108 次, 乘法操作减少了 256 倍.

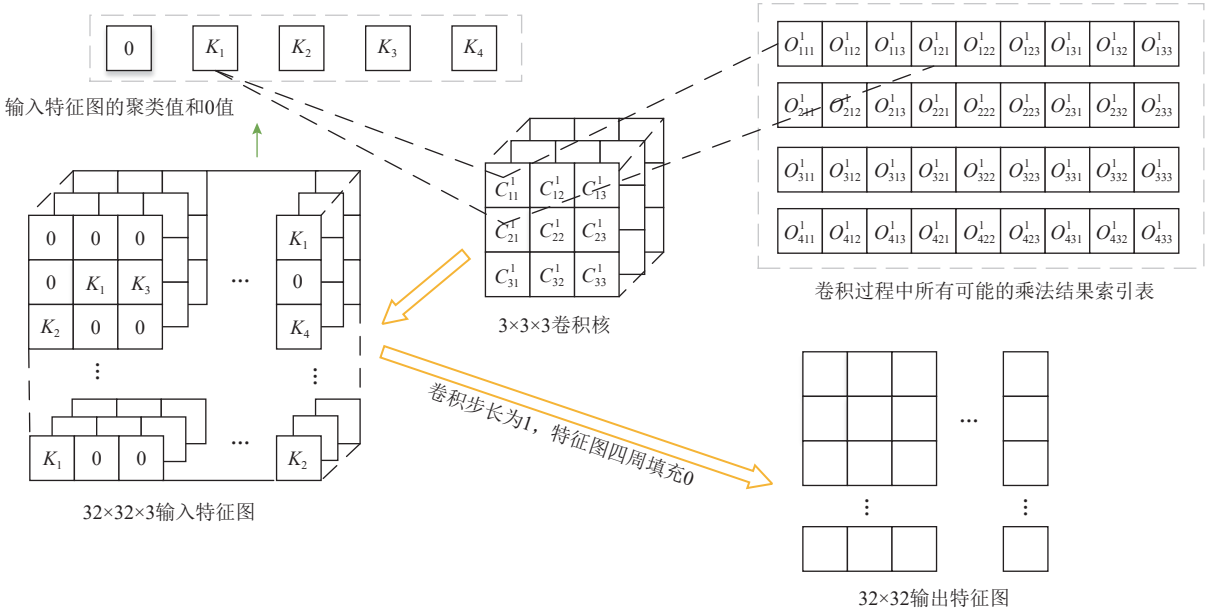


Fig. 4 A schematic diagram of the convolution process after clustering of input feature map

图 4 输入特征图聚类后的卷积流程示意图

2.3 轻量化压缩

k-means 算法除了可以应用到卷积层的加速中以外, 还可以应用到全连接层的压缩中. 如表 2 所示, 全连接层的权重数占整个网络存储的 90% 以上, 对于大规模神经网络来说这样庞大的数据不可能完全存储在片上, 这就导致需要频繁地从片外读取数据, 以往的研究表明[33], 数据搬移消耗的能量远远大于数据计算消耗的能量. 因此想要实现网络的轻量化, 全连接层权重数据压缩是非常必要的. 以往的研究已经提出许多权重压缩的方法[34-36], 其中比较常用的方法是利用全连接层的稀疏特性对全连接层的权重进行压缩编码存储, 我们经过实验证明(具体实验结果见第 4 节)对全连接层的权重数据运用 *k*-means 预处理不会损失整个网络的识别精度. 这样结合全连接层的稀疏特性, 可以进一步地压缩全连接层权重数据. 具体方法如图 5 所示, 某一个神经网络的全连接层有 10 个神经元, 与上一个全连接层的 100 个神经元进行全连接, 该全连接层则具有 1 000 个权重数

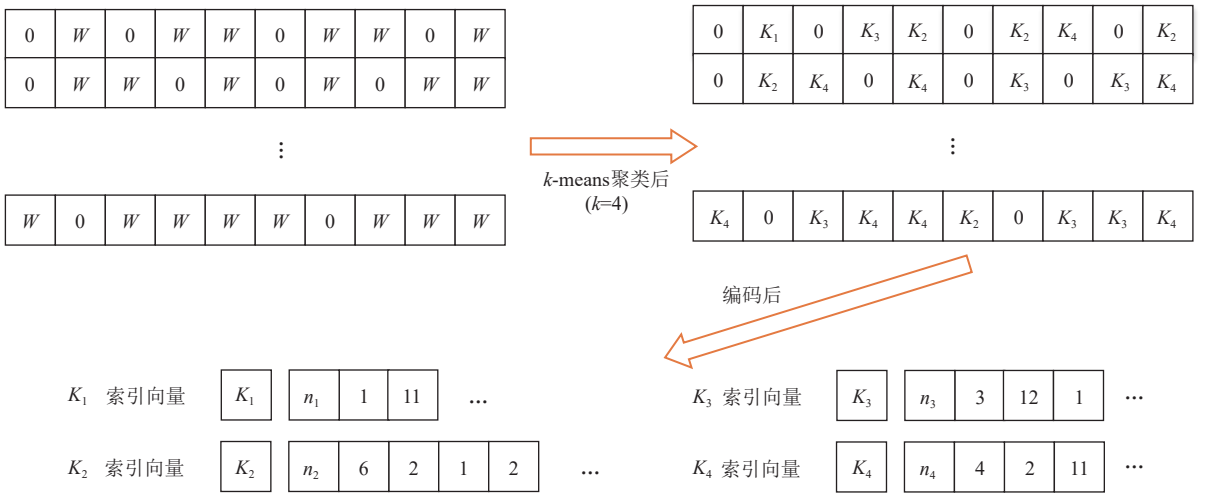


Fig. 5 Schematic diagram of the fully connected layer clustering and coding process

图 5 全连接层聚类和编码流程示意图

据，将这些权重数据的非 0 值进行 *k*-means 聚类，然后单独对每一类权重进行压缩编码，最终得到 *k* 个索引向量（*k* 为聚类种类）. 向量的第 1 个元素为该类权重的总数，后面的元素为该类权重在原来权重矩阵中的相对位置.

3 特定硬件架构 KCNN

通用处理器实现上述轻量化方法不能达到很好的加速效果，因此我们设计了一种特定的硬件加速器架构 KCNN，整体框架如图 6 所示，加速器的功能模块有 *k*-means 功能模块、乘法树功能模块、累加器功能模块、池化功能模块和激活功能模块. 加速器还设计一些缓存用于临时存储权重和神经网络计算过程中的中间值. 图 6 中红色数据流代表卷积层独有的计算过程，绿色数据流代表全连接层独有的计算过程，黑色数据流是一些共用数据流和存取数据的操作. 为了节省硬件开销，该架构的乘法树和累加器由卷积层和全连接层进行分时复用.

1）乘法树功能模块. KCNN 采用树状乘法单元结构而非更加常用的阵列结构，这是由于经过 *k*-means 聚类处理的输入特征图和卷积核在实际进行卷积处理时已经由原来的矩阵相乘运算简化为向量相乘计算，乘法单元只输入有限的聚类值，因此为了避免不必要的数据流调度，我们选择乘法树的实现方式.

2）*k*-means 功能模块. 该功能模块包含 2 个功能实现，一是判决上一层输出特征图的非 0 值，二是对这些非 0 值进行聚类. 最后将处理后的聚类值和对应的索引表存入下一层网络的特征图存储表.

3）乘法结果索引表. 乘法结果索引表里存储了卷积层所有的乘法结果和得到该乘法结果的 2 个乘数对应位置，卷积结果判别器将根据对应位置访问乘法结果存储表并把 1 次卷积操作的所有乘法结果传入累加器.

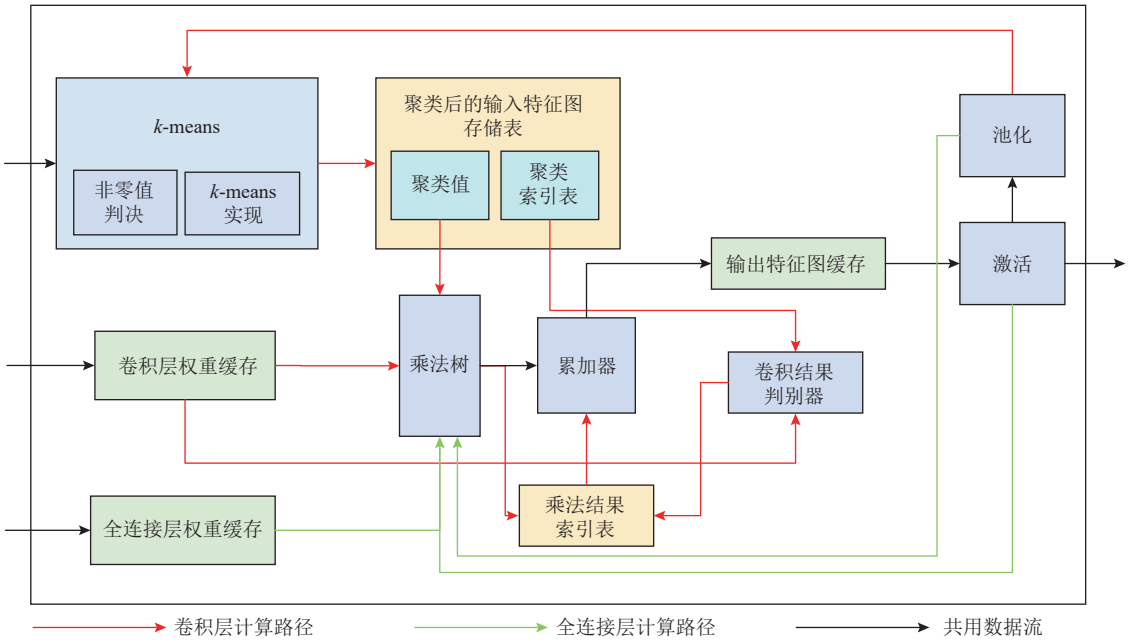


Fig. 6 The overall architecture of KCNN

图 6 KCNN 的整体架构

4）累加器功能模块. 该模块接收 1 次卷积的所有乘法结果，并累加得到输出特征图上的 1 个输出结果.

5）激活模块. 该模块提供常用的激活函数，例如 ReLu 函数和 Sigmod 函数，其输出有 3 条路径，如果下一层是卷积层，输出结果将传入 *k*-means 功能模块；如果下一层是全连接层，输出结果将传入乘法树功能模块，等待与全连接层的权重相乘；如果该激活层是网络最后一层，输出结果将作为整个神经网络的输出传到片外.

6）池化模块. 该模块实现池化功能，具有平均池化和最大池化的功能.

7）全连接层权重缓存模块. 该模块缓存经过聚类和压缩编码后的全连接层权重值和索引表.

卷积层的处理流程为：1）加速器将特征图数据输入到 *k*-means 功能模块进行非 0 值聚类，将聚类后的索引值和索引列表存入输入特征图存储表（聚类索

引表可以还原聚类值在输入特征图中的位置); 2)将卷积核权重缓存中的权重数值和聚类索引值传入乘法树进行乘法操作, 结果存入卷积乘法结果索引表; 3)将权重的位置数据和特征图聚类值的位置数据传入乘法结果判决器, 根据判决结果从卷积乘法结果索引表中取得对应的乘法结果并传入累加器; 4)将累加器得到的输出结果存入输出特征图缓存; 5)将输出特征图经过激活作为下一层网络的输入特征图. 如果下一层网络为池化层, 则将输出结果传入池化模块, 如果下一层网络为全连接层则将输出结果传入乘法树等待和全连接层的权重进行乘法操作.

全连接层的处理流程为: 1)加速器将全连接层的权重存入全连接层权重缓存, 传入乘法树中和上一层网络的输出特征图相乘; 2)将乘法结果传入累加器得到新的输出特征图; 3)将输出特征图传入激活模块得到下一层网络的输入特征图或者整个网络的输出结果.

与一般的 CNN 加速器相比, 该架构需要增加额外的硬件资源来实现权重和特征图的非 0 值聚类. 并且不同网络合适的聚类值需要提前进行大量的实验验证, 保证轻量化处理后的网络精度处于网络的置信区间, 具体的实验细节在第 4 节介绍.

4 实验评估

4.1 实验方法

实验采用 2 个主流的数据集: MNIST, CIFAR-10 和 3 个典型 CNN 结构: LeNet-5, AlexNet, VGG16. 其中 MNIST 是一个手写数字数据集; CIFAR-10 是一个常用的计算机视觉数据集, 包含 10 种不同种类的 60 000 张彩色图片. LeNet-5 是一个典型的 CNN 结构; AlexNet 是 2012 年 ImageNet 竞赛冠军获得者 Hinton 和他的学生 Alex 设计的 CNN 结构; VGG16 则是牛津大学计算机视觉组参加图像分类竞赛时提出的, 与 AlexNet 相比, 网络深度更深. 本文所有的实验在神经网络框架 Pytorch 下进行. 其中对于 *k*-means 算法我们选择 Python 机器学习算法库中的 sklearn. cluster. MiniBatchKMeans.

4.2 实验结果

4.2.1 加速效果

本文实验通过在 CNN 的卷积层前加入自定义的 *k*-means 层(*k*-means 层完成对输入特征图的非 0 值聚类, 并将聚类值放到输出特征图的对应位置), 验证选取不同的聚类 *k* 值对网络预测精确度的影响. 在网络置信区间的范围内选择最优 *k* 值并计算加速比 η. 加速比的计算公式如式(2)所示, T_{mut}和T_{add}表示不使用 *k*-means 层进行非 0 值聚类时卷积层所需的乘法操作和加法操作的时间, $T_{k\text{-means}}$表示 *k*-means 层实现非 0 值聚类所需的时间, T'_{mut}和T'_{add}表示使用 *k*-means 层进行非 0 值聚类后卷积层计算所需乘法和加法操作的时间(随 *k* 值变化).

$$\eta=\frac{T_{\text{mut}}+T_{\text{add}}}{T_{k\text{-means}}+T'_{\text{mut}}+T'_{\text{add}}}. \tag{2}$$

需要说明的是, 对于卷积层数少的网络, 例如 LeNet-5, 本文实验针对不同卷积层添加不同 *k* 值的 *k*-means 层以验证最好的加速效果, 对于卷积层数多的大型网络, 所添加的 *k*-means 层选取不同 *k* 值的组合太多, 本文并没有进行穷举, 研究者可以根据自己的需求在不同的卷积层前增加不同 *k* 值的 *k*-means 层去平衡预测精度和加速效果. 本文主要的贡献点是介绍一种新颖的轻量化方法以及实验验证该方法的可行性.

表 4 展示了 LeNet-5 不同卷积层前添加不同 *k* 值的 *k*-means 层后的网络识别准确率. 实验结果表明, LeNet-5 在 MNIST 数据集上进行预测时, 保证识别准确率在网络置信区间的前提下, 最优的 *k*-means 层添加组合为第 1 个卷积层前添加 *k*=4 的 *k*-means 层和第 2 个卷积层前添加 *k*=4 的 *k*-means 层. 此时, 整个网络的加速效果最好.

Table 4 Recognition Accuracy of Network for LeNet-5 After Adding *k*-means Layers with Different *k* Value Combinations

表 4 LeNet-5添加不同*k*值组合的*k*-means层后的识别准确率 %

卷积层	聚类值	*C*1				
		k=4	*k*=8	*k*=16	*k*=32	*k*=64
*C*2	*k*=4	99.02	99.14	99.06	99.15	99.12
	k=8	99.10	99.13	99.21	99.16	99.21
	k=16	99.12	99.16	99.19	99.20	99.16
	k=32	99.13	99.20	99.19	99.17	99.19
	k=64	99.13	99.21	99.22	99.19	99.20

注: *C*1 *k*=4 表示在第 1 个卷积层前添加聚类值为 4 的 *k*-means 层, MNIST 基准为 99.20%.

表 5 展示了 AlexNet 不同卷积层前单独添加不同 *k* 值的 *k*-mean 层后网络在 CIFAR-10 数据集上的识别准确率.实验结果表明, 在 AlexNet 的 5 个不同卷积层前单独添加合适 *k* 值的 *k*-means 层后, 网络的识别准确率仍处于网络的置信区间. VGG16 具有 13 个卷积层, 由于卷积层数太多, 因此我们设计的实验方

法为在所有的卷积层前添加相同 k 值的 k-means 层，实验结果见表 6. 其中，VGG16 和 AlexNet 所有卷积层前都添加 k=32 或 k=64 的 k-means 层时，网络的识别准确率处于置信区间内. 表 5 和表 6 表明，对于深度 CNN，该轻量化方法依然有效. 开发者可以根据实际使用的神经网络和数据集，灵活地添加不同 k 值的 k-means 层，本文不再对所有的组合进行穷举.

Table 5 Recognition Accuracy of Network for AlexNet After Adding Different *k*-means Layers with Different *k* Values

表 5 AlexNet添加不同*k*值*k*-means层后的识别准确率 %

聚类值	*C*1	*C*2	*C*3	*C*4	*C*5
k=4	66.60	81.28	81.70	82.23	82.80
k=8	78.57	82.67	82.66	82.76	83.06
k=16	81.92	82.82	82.79	82.77	82.98
k=32	82.56	82.96	82.79	82.97	82.98
k=64	82.97	82.93	82.96	82.86	82.87

注：*C*1 表示在第 1 个卷积层前添加 k-means 层，k=4 表示添加聚类值为 4 的 k-means 层，CIFAR-10 基准为 82.33%.

Table 6 Recognition Accuracy on CIFAR of VGG16 and AlexNet After Adding *k*-means Layers with the Same *k* Value

表 6 VGG16 和 AlexNet 所有的卷积层前添加相同 *k* 值的 *k*-means 层后在 CIFAR-10 上的识别准确率 %

CNN（数据集）	基准	k=4	k=8	k=16	k=32	k=64
AlexNet（CIFAR-10）	82.33	61.28	77.34	81.47	82.67	82.84
VGG16（CIFAR-10）	91.30	61.58	82.91	88.70	90.81	91.45

4.2.2 压缩效果

不同网络采用 k-means 进行全连接层聚类后的压缩效果不同，具体的压缩比 γ 可以通过式(3)计算得出，N 表示所有权重的数量，N_c 表示卷积层的权重数目，b 表示权重的位数(论文取 float32)，N_f' 表示全连接层非 0 权重的数量，b_{index} 表示非 0 权重的索引值位数(由相同 2 个聚类值在权重矩阵中的最大间距计算得出)，k 表示聚类值个数.

$$\gamma=\frac{N\times b}{N_c\times b+k\times b+N_f'\times b_{index}}. \tag{3}$$

本节只验证 CNN 的全连接层经 k-means 非 0 聚类压缩后是否对最终的网络推导准确率造成影响. 我们对网络的全连接层采用不同的 k 值进行压缩，对于有多个全连接层的网络来说，同样存在每个层选择不同聚类 k 值的多种组合情况. 我们在选择不同的 k 值组合时，应保证最终的识别准确率处于网络置信区间内.

表 7 显示了 3 个不同网络在不同数据集上的全连接层的最优 k 值组合和最终准确率. 实验结果表明本文提出的轻量化方法对网络的权重压缩有效，且不损失网络的识别准确率.

Table 7 Recognition Accuracy of the Network After Clustering Compression on Fully Connected Layers

表 7 全连接层经过聚类压缩后的网络识别准确率

CNN（数据集）	基准/%	k 值组合	识别准确率/%
LeNet-5（MNIST）	99.20	(8, 4, 32)	99.03
AlexNet（CIFAR-10）	82.33	(4, 4, 32)	81.43
VGG16（CIFAR-10）	91.30	(4, 4, 2)	90.72

注：(8, 4, 32) 表示 3 个全连接层权重分别进行聚类值 k=8, 4, 32 的聚类处理.

本文对 AlexNet 网络的进行了完整的轻量化处理，实验表明，在 AlexNet 的卷积层前分别添加 k=32, 32, 8, 8, 8 的自定义 k-means 层，再对全连接层权重分别进行 k=4, 4, 32 的聚类处理，此时网络在 CIFAR-10 上 81.94% 的识别准确率仍处于置信区间，网络的加速比为 3，压缩比为 6.6. 需要注意的是，CNN 在处理不同的数据集时，最优的聚类值组合会不同，在将 CNN 部署在加速器前，应该提前计算好最优的 k 值组合.

5 结　论

本文提出了一种新颖的 CNN 轻量化方法和硬件实现架构，创造性地发掘并利用 CNN 推导过程中的重复数据，利用在卷积层前添加自定义的 k-means 层来实现更多的非 0 值数据复用，极大地减少了计算开销，通过对全连接层的权重进行 k-means 非 0 聚类实现网络权重的压缩，极大地减少了存储的开销. 不同于以前的轻量化方法，该方法不需要进行重新训练，可以直接利用训练好的网络结构和权重数据进行优化. 最终该方法可以减少 AlexNet 卷积层 66% 的计算量，同时实现 6.6 倍的压缩比. 本文还设计了针对该轻量化方法的硬件架构 KCNN，对于该架构的详细性能和功耗评测将是我们未来的工作方向.

作者贡献声明：陈桂林负责论文的创新点设计和实验；王观武负责论文的实验指导和创新点优化；王康负责论文相关技术的背景现状梳理；胡敏慧负

责论文的创新点探讨和校稿;邓俊荃负责论文的框架指导和内容梳理.

参 考 文 献

[1] Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks[J]. Advances in Neural Information Processing Systems, 2012, 25(2): 1097−1105

[2] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM networks[C]//Proc of the IEEE Int Joint Conf on Neural Networks. Piscataway, NJ: IEEE, 2005: 2047−2052

[3] Conneau A, Schwenk H, Barrault L, et al. Very deep convolutional networks for text classification[C]//Proc of the 15th Conf of the European Chapter of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2017: 1107−1116

[4] Zhang Jiaqi, Chen Xiangru, Song Mingcong, et al. Eager pruning: Algorithm and architecture support for fast training of deep neural networks[C]//Proc of the 46th Annual Int Symp on Computer Architecture (ISCA). New York: ACM, 2019: 292−303

[5] Yu J, Lukefahr A, Palframan D, et al. Scalpel: Customizing DNN pruning to the underlying hardware parallelism[C]//Proc of the 44th Int Symp on Computer Architecture (ISCA). New York: ACM, 2017: 548−560

[6] Sharify S, Lascorz A D, Mahmoud M, et al. Laconic deep learning inference acceleration[C]//Proc of the 46th Int Symp Computer Architecture (ISCA). New York: ACM, 2019: 304−317

[7] Wang Naigang, Choi J, Brand D, et al. Training deep neural networks with 8-bit floating point numbers[C]//Proc of the 32nd Int Conf on Neural Information Processing Systems (NIPS'19). New York: ACM, 2019: 7686−7695

[8] Hoefler T, Dan A, Ben-Nun T, et al. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks[J]. arXiv preprint, arXiv: 2102: 00554, 2021

[9] Ding Caiwen, Liao Siyu, Wang Yanzhi, et al. CirCNN: Accelerating and compressing deep neural networks using block-circulant weight matrices[C]//Proc of the 50th Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2017: 395–408

[10] Guo Cong, Zheng Chen, Leng Jingwen, et al. ANT: Exploiting adaptive numerical data type for low-bit deep neural network quantization[C]//Proc of the 55th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2022: 1414−1433

[11] Parashar A, Rhu M, Mukkara A, et al. SCNN: An accelerator for compressed-sparse convolutional neural networks[C]//Proc of the 44th Annual Int Symp on Computer Architecture. Los Alamitos, CA: IEEE Computer Society, 2017: 27−40

[12] Wu Xinxin, Ou Yan, Li Wenming, et al. Acceleration of sparse convolutional neural network based on coarse-grained dataflow architecture[J]. Journal of Computer Research and Development, 2021, 58(7): 1504−1517 (in Chinese)
(吴欣欣, 欧焱, 李文明, 等. 基于粗粒度数据流架构的稀疏卷积神经网络加速[J]. 计算机研究与发展, 2021, 58(7): 1504−1517)

[13] Han Song, Liu Xingyu, Mao Huizi, et al. EIE: Efficient inference engine on compressed deep neural network[C]//Proc of the 43rd Int Symp on Computer Architecture. New York: ACM, 2016: 243−254

[14] Chen Yunji, Chen Tianshi, Du Zidong, et al. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning[C]//Proc of the 19th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2014: 269−284

[15] Du Zidong, Fasthuber R, Chen Tianshi, et al. ShiDianNao: Shifting vision processing closer to the sensor[C]//Proc of the 42nd Annual Int Symp on Computer Architecture. New York: ACM, 2015: 92–104

[16] Farabet C, Martini B, Corda B, et al. NeuFlow: A runtime reconfigurable dataflow processor for vision[C]//Proc of the 29th Computer Vision and Pattern Recognition Workshops. Piscataway, NJ: IEEE, 2011: 109−116

[17] Chen Y H, Emer J, Sze V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks[C]//Proc of the 43rd Annual Int Symp on Computer Architecture. New York: ACM, 2016: 367–379

[18] Sankaradas M, Jakkula V, Cadambi S, et al. A massively parallel coprocessor for convolutional neural networks[C]//Proc of the 20th IEEE Int Conf on Application-Specific Systems, Architectures and Processors. Piscataway, NJ: IEEE, 2009: 53−60

[19] Sriram V, Cox D, Tsoi K H, et al. Towards an embedded biologically-inspired machine vision processor[C]//Proc of the 9th Int Conf on Field-Programmable Technology. Piscataway, NJ: IEEE, 2011: 273−278

[20] Chakradhar S, Sankaradas M, Jakkula V, et al. A dynamically configurable coprocessor for convolutional neural networks[C]//Proc of the 38th Int Symp on Computer Architecture. New York: ACM, 2010: 247−257

[21] Edward H, Li Shiyu, Hai H L, et al. Cascading structured pruning: Enabling high data reuse for sparse DNN accelerators [C]//Proc of the 49th Annual Int Symp on Computer Architecture (ISCA 2022). New York: ACM, 2022: 522−535

[22] Li Gang, Xu Weixiang, Song Zhuoran, et al. Ristretto: An atomized processing architecture for sparsity-condensed stream flow in CNN[C]//Proc of the 55th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2022: 1434−1450

[23] Chen Guilin, Ma Sheng, Guo Yang. Survey on accelerating neural network with hardware[J]. Journal of Computer Research and Development, 2019, 56(2): 240−253(in Chinese)
(陈桂林, 马胜, 郭阳. 硬件加速神经网络综述[J]. 计算机研究与发展, 2019, 56(2): 240−253)

[24] Ji Rongrong, Lin Shaohui, Chao Fei, et al. Deep neural network compression and acceleration: A review[J]. Journal of Computer Research and Development, 2018, 55(9): 1871−1888 (in Chinese)
(纪荣嵘, 林绍辉, 晁飞, 等. 深度神经网络压缩与加速综述[J]. 计算机研究与发展, 2018, 55(9): 1871−1888)

[25] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit[C]//Proc of the 44th Annual Int

Symp on Computer Architecture. New York: ACM, 2017: 1−12

[26] Reagen B, Whatmough P, Adolf R, et al. Minerva: Enabling low-power, highly-accurate deep neural network accelerators[C]//Proc of the 43rd Annual Int Symp on Computer Architecture (ISCA). New York: ACM, 2016: 267−278

[27] Chen Guilin, Wang Guanwu, Ju Jian, et al. Research on the influence of k-means cluster preprocessing on adversarial images[C]//Proc of the 7th Int Conf on Information Technology. New York: ACM, 2019: 248−252

[28] Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks[J]. arXiv preprint, arXiv: 1312.6199, 2013

[29] Xu Dongkuan, Tian Yingjie. A comprehensive survey of clustering algorithms[J]. Annals of Data Science, 2015, 2(2): 165−193

[30] Macqueen J. Some methods for classification and analysis of multivariate observations[C]//Proc of the 5th Berkeley Symp on Mathematical Statistics and Probability. Oakland, CA: University of California, 1966: 281−297

[31] Chen Guilin, Ma Sheng, Guo Yang. Towards convolutional neural network acceleration and compression via *k*-means cluster[J]. Computer Engineering & Science, 2019, 41(5): 796−803 (in Chinese)
(陈桂林, 马胜, 郭阳, 等. 通过 *k*-means 算法实现神经网络的加速和压缩[J]. 计算机工程与科学, 2019, 41(5): 796−803)

[32] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. Handbook of Systemic Autoimmune Diseases, 2009, 1(4): 1−60

[33] Li Zhen, Wang Yuqing, Zhi Tian, et al. A survey of neural network accelerators[J]. Frontiers of Computer Science, 2017, 11(5): 746−761

[34] Yang Zichao, Moczulski M, Denil M, et al. Deep fried convnets[C]// Proc of the IEEE Int Conf on Computer Vision (ICCV). Los Alamitos, CA: IEEE Computer Society, 2015: 1476−1483

[35] Denton E, Zaremba W, Bruna J, et al. Exploiting linear structure within convolutional networks for efficient evaluation[C]//Proc of the 27th Int Conf on Neural Information Processing Systems (NIPS'14). New York: ACM, 2014: 1269–1277

[36] Srinivas S, Babu R V. Data-free parameter pruning for deep neural networks[J]. arXiv preprint, arXiv: 1507.06149, 2015



Chen Guilin, born in 1994. PhD candidate, engineer. His main interests include architecture, and Chiplet and neural network accelerator. (cglnudt@163.com)

陈桂林, 1994 年生. 博士研究生, 工程师. 主要研究方向为体系结构、芯粒和神经网络加速器.



Wang Guanwu, born in 1986. PhD, assistant professor. His main research interests include architecture and system-on-chip design.

王观武, 1986 年生. 博士, 助理研究员. 主要研究方向为体系结构、片上系统设计.



Wang Kang, born in 1990. Master, assistant professor. His main research interests include architecture and system-on-chip design.

王　康, 1990 年生. 硕士, 助理研究员. 主要研究方向为体系结构、片上系统设计.



Hu Minhui, born in 1994. PhD. Her main research interests include artificial intelligence and computer architecture.

胡敏慧, 1994 年生. 博士. 主要研究方向为人工智能、计算机体系结构.



Deng Junquan, born in 1988. PhD, associate research fellow. His main research interest includes radio resource management, mobile positioning, and machine learning in wireless networks.

邓俊荃, 1988 年生. 博士, 副研究员. 主要研究方向为无线网络的射频资源管理、移动定位和机器学习.