# 深度学习中的内存管理问题研究综述

**马玮良[1,2]，彭轩[1,2]，熊倩[1,2]，石宣化[1,2]，金海[1,2]**
1. 华中科技大学计算机科学与技术学院，湖北 武汉 430074；
2. 华中科技大学大数据技术与系统国家地方联合工程研究中心，服务计算技术与系统教育部重点实验室，湖北 武汉 430074

**摘要**
近年来，深度学习已经在多个领域取得了巨大的成功。深度神经网络向着更深更广的方向发展，训练和部署深度神经网络模型都将面对巨大的内存压力。加速设备有限的内存空间已经成为限制神经网络模型快速发展的重要因素，如何在深度学习中实现高效的内存管理成为深度学习发展的关键问题。为此，介绍了深度神经网络的基本特征；分析了深度学习训练过程中的内存瓶颈；对一些代表性的研究工作进行了分类阐述，并对其优缺点进行了分析；对深度学习中内存管理技术的未来发展趋势进行了探索。

**关键词**
内存管理；深度学习；内存交换；重计算；内存共享；压缩

中图分类号：TP183 文献标识码：A doi: 10.11959/j.issn.2096-0271.2020033

# *Memory management in deep learning: a survey*

**MA Weiliang[1,2], PENG Xuan[1,2], XIONG Qian[1,2], SHI Xuanhua[1,2], JIN Hai[1,2]**
1. School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
2. National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Huazhong University of Science and Technology, Wuhan 430074, China

***Abstract***
In recent years, deep learning has achieved great success in many fields. As the deep neural network develops towards a deeper and wider direction, the training and inference of a deep neural network face huge memory pressure. The limited memory space of accelerating devices has become an important factor restricting the rapid development of deep neural network. How to achieve efficient memory management in deep learning has become a key point in the development of deep learning. Therefore, the basic characteristics of deep neural network were introduced firstly and memory bottleneck in deep learning training was analyzed. Some representative research works were classified, and their advantages and disadvantages were analyzed. Finally, some important direction and tendency of memory management in deep learning were suggested.

***Key words***
memory management, deep learning, memory swapping, recomputation, memory sharing, compression

# 1 引言

互联网规模的迅速扩张促使全球数据呈现爆炸式增长、海量聚集的特点，社交媒体、物联网等技术的迅速发展导致了大量非结构化数据的出现，从海量数据中提取有价值的信息的难度越来越大[1]。在大数据时代，深度神经网络（deep neural network，DNN）借助大规模数据的训练取得了极高的准确率，深度学习得以快速发展，并逐渐应用于人们生活的方方面面。

近年来，人们见证了深度神经网络在许多领域取得的成功，如计算机视觉、语音识别、自然语言处理等。这些成功是由深度神经网络新架构的创新带来的。卷积神经网络（convolutional neural network，CNN）对空间模式进行建模，在计算机视觉任务中能够达到当前最优的预测结果[2-4]；循环神经网络（recurrent neural network，RNN）在序列建模和结构预测方面也取得了令人备受鼓舞的结果[5]。深度和宽度是构建深度神经网络最重要和有效的两个因素[2,4,6-8]。神经网络的深度越深、功能层越多，越能有效地降低超参数选择的复杂性，提高模型的鲁棒性。与深度相比，宽度是构建网络的另一个重要因素，它通过不同大小的卷积积累了更多的特征图。

深度学习应用是一种计算密集型和内存密集型的任务。图形处理器（graphic processing unit，GPU）、专用集成电路（application specific integrated circuit，ASIC）、可编程逻辑门阵列（field-programmable gate array，FPGA）、张量处理器（tensor processing unit，TPU）等各种专用加速设备为深度神经网络的发展提供了强大的算力支撑。设计更深层次的深度神经网络可以达到更高的精度，但是这也给各种加速设备带来了极大的挑战。例如，谷歌公司提出的基于转换器的双向编码表征（bidirectional encoder representations from transformers，BERT）模型[9]在训练中使用了768个隐藏层，占用了73 GB的内存（批处理大小为64）。然而，高带宽的GPU内存是一种稀缺资源，目前比较强大的NVIDIA GPU V100内存最多只有32 GB，而主流商用云计算GPU类型（如P100）只有16 GB内存。这一限制阻碍了深度学习研究者去探索更先进的模型架构。目前，有一些工作探讨了大数据环境下新型存储系统的相关内容，从存储系统的角度来解决大数据时代的内存瓶颈问题。王孝远等人[10]从体系结构、系统软件等多方面对当前面向大数据的异构内存系统进行了分析研究，提出了一系列异构内存系统的优化方法；陈游旻等人[1]详细阐述了构建大数据环境下的存储系统所面临的挑战、当前的研究方向以及未来的发展趋势；李鑫等人[11]则从大数据应用的角度对混合存储架构进行了深入的探讨。本文从深度学习应用的角度，侧重于探讨深度学习系统中的内存管理问题，与之前的工作有本质的不同。

内存管理是大规模深度学习发展的一个重要挑战。深度学习中的内存管理已经成为当前深度学习系统研究的重要问题。本文将介绍深度神经网络的基本特征以及训练过程，分析深度学习中内存管理的问题，从技术的角度对一些代表性工作进行分类阐述，对比它们的优缺点，并对深度学习中内存管理的未来发展趋势进行展望。

# 2 背景介绍

## 2.1 DNN的结构及训练过程

DNN是由多种不同类型的层组成的层

次结构模型，例如，用于计算机视觉任务的卷积神经网络、用于自然语言处理的循环神经网络都可以被称为深度神经网络。

笔者将通过一个典型的卷积神经网络对DNN的模型结构进行具体说明，如**图1**所示。神经网络需要经过训练才能用于推理或分类任务。训练通过执行正向传播（forward propagation）算法和反向传播（backward propagation）算法[12]，学习和更新神经网络各层的权值。

对于正向传播和反向传播来说，遍历的方向以及必须执行的操作是不同的。正向传播从第一层执行到最后一层，而反向传播从相反的方向执行（从最后一层到第一层）。正向传播遍历了整个神经网络层，并针对给定的输入执行特征提取和分类任务，从而完成图像分类。在正向传播过程中，每一层的数学操作应用于其输入特征图$X$，将计算结果进行保存，并作为输出特征图$Y$。对于线性神经网络来说，第$N-1$层的输出结果$Y$直接用作第$N$层的输入$X$，如**图1**所示。因此正向传播的计算是一个序列化的过程，第$N$层只有在第$N-1$层完成计算并将其输出结果$Y$传递到第$N$层的输入$X$时，才能开始相应的操作。

对于未经充分训练的DNN来说，推断的图像类别可能是不准确的。因此，笔者使用一个损失函数来推导正向传播结束时推理误差的大小。具体来说，损失函数的梯度是根据最后一层的输出值推导出来的：

$$\mathrm{d}Y=\frac{\partial \mathrm{Loss}}{\partial Y_{(N)}} \tag{1}$$

由式（1）可以得到最后一层的输出结果的梯度$\mathrm{d}Y$，进而根据链式法则[12]，能够得到最后一层输入的梯度$\mathrm{d}X$：

$$\mathrm{d}X=\frac{\partial \mathrm{Loss}}{\partial X_{(N)}}=\frac{\partial \mathrm{Loss}}{\partial Y_{(N)}}\cdot\frac{\partial Y_{(N)}}{\partial X_{(N)}} \tag{2}$$

由式（2）可以看出，计算第$N$层的输入$X$的梯度值$\mathrm{d}X$需要的内存空间包括输入/输出梯度映射（$\mathrm{d}Y$和$\mathrm{d}X$）的内存空间和该层的输入/输出特征映射（$X$和$Y$）的内存空间。对于线性网络，将计算得到的第$N$层的$\mathrm{d}X$直接传递到第$N-1$层，作为第$N-1$层$\mathrm{d}X$推导的$\mathrm{d}Y$。类似地，这个链式法则被用来推导权值的梯度，从而更新网络模型。与正向传播类似，反向传播也对各个梯度映射按层执行。当反向传播到达第一层时，利用权值梯度调整权值，以减少下一个分类任务的预测误差。

## 2.2 深度学习中的内存管理问题

近年来，机器学习框架如雨后春笋般蓬勃发展，如TensorFlow[13]、Theano[14]、PyTorch[15]和MxNet[16]等。这些框架提供

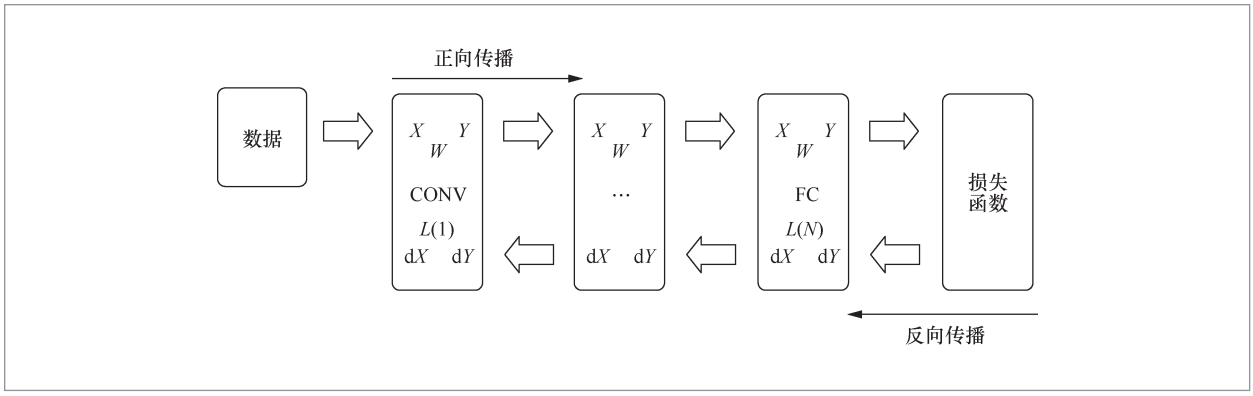


图1 CNN模型的训练过程

了丰富的特性来帮助开发者快速设计各种各样的神经网络，并且使用各种加速设备来加速DNN的训练和推理，极大地简化了神经网络的实现过程，成为帮助研究者开展研究的利器。但是这些机器学习框架在内存管理方面受到严重的限制。

现有的机器学习框架在训练DNN时，DNN所有层所需的内存空间必须能够被放在加速设备的内存中，以此来获得性能上的优势。由于层与层之间存在数据依赖关系，加速设备在任何给定时间都只能处理单层的计算。因此，不管神经网络的结构如何，神经网络在训练过程中的计算都是分层进行的。整个神经网络的内存分配策略没有考虑到DNN分层训练的特点，这对内存资源造成了极大的浪费。训练一个神经网络需要经过数百万甚至数亿次的迭代才能达到理想的训练效果。由于基于随机梯度下降（stochastic gradient-descent，SGD）的反向传播算法具有随机性[12]，神经网络的输入通常成批地训练数据，这也将显著增加内存占用，但是有助于网络模型更好地收敛到最优解。随着DNN的网络结构向着更深、更广的方向发展，训练DNN时所需的内存空间越来越大，单个加速设备的内存已经不能满足训练的需求。

包括DistBelief[17]等在内的一些系统试图分割神经网络模型，从而在多个GPU上进行分布式训练。这种类型的训练方式通常被称为模型并行，其可以显著地缓解单个加速设备的内存压力。然而，由于多GPU之间需要频繁地通信来更新模型参数，导致它的性能较差。

因此，需要通过内存管理解决现有内存分配策略带来的内存浪费问题，使得深度学习系统能够充分利用加速设备的算力和有限的内存资源，保证大规模神经网络在单个加速设备中能够快速训练。

在训练过程中，加速设备的内存主要被3个部分消耗：存储正向传播中产生的特征映射、存储反向传播中的梯度映射以及卷积算法需要的额外内存空间[18]。与之相比，模型的权值消耗的内存非常少，并且其通常在加速设备内存中长期存在，可以持续更新。在这3个部分中，后两个部分是临时内存，可以在当前计算完成后立即释放。正向传播和反向传播的计算都需要特征映射，只有在反向传播计算完成后，对应层的特征映射占用的内存空间才能够被释放。参考文献[18]中也指出特征映射和卷积算法所需的缓存空间占据着主要的内存空间。因此，降低特征映射的内存消耗成为目前大多数工作的主要目标，而特征映射在正向传播和反向传播中参与的两次计算之间存在很大的时间间隔，这也为内存管理带来很多可能。

经过训练的神经网络模型常常被部署在边缘计算设备中。在边缘计算设备中执行神经网络模型的推理阶段具有隐私保护和实时处理的优势，但是边缘计算设备往往具备更少的计算和内存资源，并且部分设备的能源是有限的，对应用能耗的要求很高。与训练阶段不同，在推理阶段，神经网络模型只执行正向传播的计算过程。因为没有反向传播过程来更新模型权值，所以正向计算过程中的中间结果不需要保存。因此，在推理阶段，计算设备的内存主要被模型的权值以及计算时所需的临时内存空间消耗。而这些内存需求已经给边缘计算设备带来了较大的内存开销。因此，在推理阶段进行内存管理也是必要的。优化模型结构是当前研究的主要方向。优化模型结构一方面能够通过剔除一些不必要的模型权值来降低模型的大小；另一方面能够优化层与层之间的连接，从而减少计算过程。通过优化模型结构能够有效地降低推理过程中的内存消耗。

## 3 深度学习中内存管理的关键技术

目前已经有很多工作致力于降低单个设备训练神经网络时的内存消耗。笔者从深度学习中内存管理用到的技术的角度对当前一些代表性工作进行分类阐述，并对其优缺点进行分析。

### 3.1 内存交换

内存交换技术是指在加速设备内存和主存之间交换数据，通过在不使用变量时将其从加速设备的内存交换到主存的方式来降低加速设备的内存消耗，并在下一次访问变量之前将其交换回加速设备内存。加速设备的内存容量相对于主存来说要小很多。例如，目前比较新的NVIDIA GPU V100最大只有32 GB的内存，而服务器中主存的大小通常在100 GB左右，甚至更大。内存交换能够交换几乎所有的设备内存数据，因此其能够大幅度地降低设备内存的占用率。在理想情况下，数据在主存和设备内存之间的通信可以隐藏在计算之下，从而最小化数据传输开销。

Rhu等人[18]提出了一个运行时内存管理解决方案，并将其称为虚拟深度神经网络（virtualized deep neural network，vDNN）。vDNN在主存和GPU内存之间虚拟化了神经网络的内存使用。特征映射作为中间数据结构长期驻留在GPU内存中，并且消耗了大量的GPU内存。因此，vDNN在正向传播时，将特征提取层的输出结果特征映射交换至主存中，从而降低GPU内存的占用；在反向传播时，及时地将其交换至GPU内存中参与梯度计算。为了减少传输时间，vDNN将计算与数据传输并发进行，以此隐藏数据传输的时间。但是，vDNN在每一层的末端同步数据计算和数据传输，也就是说，只有在计算和传输都完成之后，它才会继续下一层的计算。这可能导致GPU在开始下一层计算之前必须等待数据传输完成。数据在主存和GPU内存之间的传输时间并不能完全与GPU的计算时间重叠，笔者还观察到，卷积（CONV）层相对于激活（ACTV）层和池化（POOL）层有更长的计算时间。为了降低内存交换带来的性能损失，vDNN也提供了只将CONV层的特征映射交换至主存的策略，这样就有更多的时间来隐藏由交换内存带来的通信时延。vDNN仅适用于分层的卷积神经网络。

TensorFlow使用一个统一的数据流图来表示一个模型训练任务。图中的节点表示计算。在节点之间的边则保存张量信息。每个节点都由执行程序调度执行。可以将这个图看作训练任务的中间表示，因此对于模型来说，对这个图的优化是通用的和透明的。Chen等人[19]通过修改数据流图的方式实现了内存交换的策略。根据对数据流图的分析，选择生命周期较长的特征映射作为交换到主机内存的目标，这有助于降低通信带来的开销。此外，过早地将其交换进来会加剧GPU内存的占用，而过晚地将其交换进来又会带来性能开销，因此需要选择合适的触发器来将交换出去的特征映射交换至GPU内存中。Chen等人评估了数据流图中每个节点的计算时间，以此为基础为每一个目标特征映射选择合适的触发器。优化后的数据流图可以直接在TensorFlow中运行。该方法适用于所有的神经网络模型。

Chen等人[20]应用数据卸载和数据预取的思想，使用启发式的调度算法完成数据传输。此外作者提到不同的CONV算法计算所需的内存空间大小不同，因此提出了动态选择CONV算法，根据当前设备内

存的占用情况选择合适的CONV算法，最终使得内存占用和性能都得到优化。Jin等人[21]提出了一种基于CPU-GPU数据迁移的层间内存复用策略，作者记录了在GPU中每一层数据的内存访问顺序，并跟踪被其他层引用的数据的引用计数。通过这种方式，可以建立训练过程的细粒度内存访问序列，以确定何时迁移数据。作者提出的方法能够有效地降低内存交换带来的性能损失。

Wang等人[22]创建了一个统一的张量池，实现了对张量的分配和释放，同时能够将一些张量交换至主存中，以降低GPU内存的占用。此外，运行时系统会尽可能地将通信与计算重叠，以保证训练性能。然而，计算的时间是固定的，因此这种重叠的机会是有限的。作者在GPU内存上创建了一个张量缓存，通过张量复用来最小化总通信开销。

Peng等人[23]提出了基于张量的GPU内存管理（tensor-based GPU memory management）方案——Capuchin。作者观察到，所有机器学习框架都基于数据流图执行DNN模型，其中处理过程是基于张量的运算。深度学习训练中的张量访问表现出数据重用和固定访问模式的特征。此外，一次训练过程通常由数百万次的迭代组成。作者还观察到，张量访问模式具有跨迭代的规律和重复访问的特点，通过分析时序和张量访问模式可以实现高效的内存管理。Capuchin通过动态分析模型训练时的张量访问特征，在主存和GPU内存之间精确地执行张量的卸载和预取操作，在保证训练性能的同时大幅降低了DNN模型训练时的内存占用。

通过对神经网络训练过程中一些特征的观察，许多工作采用内存交换的方式进行训练时的内存管理，使得有限的GPU内存能够充分发挥作用，满足更深、更广的神经网络模型的训练需要。但是如何降低内存交换产生的性能损失还激励着研究者们不断探索新的内存交换策略，最终实现在保证训练性能的同时大幅降低设备内存占用的目的。

### 3.2 重计算

重计算技术的思想是将特征映射这样的中间结果在正向传播过程中及时地释放，在反向传播的计算需要用到特征映射时，再通过重新计算的方式生成，进而参与到当前计算中。这是一种利用计算来换取内存空间的思想。目前有一些工作采用这种方式进行神经网络训练时的内存管理，他们通常将计算成本低的中间结果作为重计算的目标，在保证训练性能的同时尽可能地减少内存消耗。

Chen等人[24]提出了一种Gradient-Checkpoint算法。具体地说，作者将神经网络划分成几个部分，算法只保留每部分的输出结果，删除了所有中间结果。在反向传播期间，如果需要用到被删除的结果，则根据每一部分保留的信息重新进行计算。该算法需要的内存空间包括存储每一部分输出结果的内存空间和每一部分反向传播计算时所需的最大内存空间，其通过一次额外的正向计算降低占用的内存。作者在该算法的基础上，对如何进行神经网络的划分以及选取重计算的目标数据做了进一步的探讨，最终实现了通过为每层增加一个额外的正向计算，即可训练一个只有$O\sqrt{n}$内存开销的$N$层线性深度神经网络的目的。

Gruslys等人[25]将重计算的思想与基于时间的反向传播（backpropagation through time）算法结合，提出了一种在训练RNN时减少内存消耗的方法。该方法通过动态编程来平衡中间结果的缓存和重计

算，能够在用户指定的任何内存约束下完成神经网络的训练。但是Chen等人[24]和Gruslys等人[25]提出的方案都基于对线性计算图的一个强力假设：计算图中所有中间变量的内存开销都是相同的，因此他们的工作都局限于一些特定的神经网络中。通常，有以速度为中心和以内存为中心的重计算策略。以速度为中心的策略保留了重新计算的张量，以便反向计算时其他层也可以直接重用它们。以内存为中心的策略总是重新计算每个反向计算层依赖的张量，与以速度为中心的策略不同，它会释放重新计算的中间结果，充分利用了节省内存的机会。Wang等人[22]提出的Cost-Aware重计算方法充分利用了这两种策略的优势，确保最高的内存占用情况与以内存为中心的策略保持一致，而额外的计算开销则与以速度为中心的策略相当。

Kusumoto等人[26]提出了一种新的有效的重计算方法，该方法可以应用于更广泛的神经网络模型。作者使用图论的语言将在固定的内存大小约束下最小化计算开销的一般重计算问题形式化，并提供了一种动态规划的解决方案。

基于对张量访问模式的分析，Capuchin能够根据当前内存的占用情况动态地选取重计算的张量，并决定张量的重计算时间，有效地降低了重计算带来的性能开销。Chen等人[24]提出的Checkpointing方案假设计算图中所有节点的内存开销是相同的，并且梯度是无法被重构的。Jain等人[27]注意到这些假设限制了先前方法的效率和通用性，因此，他们提出了Checkmate方案，将DNN训练时间与训练时内存需求之间的权衡问题抽象为张量重构的优化问题，并将其形式化为混合整数线性规划问题，根据求得的结果选择可用于重构的张量，进而制定高效的重计算策略。Checkmate是对Checkpointing策略的一般化，适用于任意类型的深度神经网络。

任何重计算方法的有效性都取决于其定义的规则：缓存哪些变量以及如何重计算其他变量。目前的研究者围绕这一问题不断提出新的方法，以期望用最小的性能开销换取最大的内存空间。

### 3.3 内存共享

内存共享技术指的是通过对不同变量生命周期的分析，在不同变量之间重复使用同一块内存空间。在机器学习框架中有两种类型的内存共享方式：置换操作和内存复用。置换操作是将输出结果直接存储在输入数据的物理地址上。例如，当计算$y$=sigmod($a$)时，$y$可以直接存储在$a$的内存中。内存复用则是指在生命周期不重叠的变量之间共享同一块内存。

Chen等人[24]也应用了内存共享的思想。作者构造了以每个变量为节点的冲突图，然后按拓扑顺序遍历整个图，并使用计数器来表示节点的生命周期。如果当前操作的输入变量没有被其他操作引用，那么当前操作的输出变量就可以使用置换操作。当节点的生命周期没有重叠时，节点之间就会发生内存复用。此外作者使用静态内存分配算法，在训练开始之前将内存分配给每个节点，以避免运行时的垃圾回收开销。

Jin等人[21]在层内以及层间应用了内存复用的策略。作者提出了一种层内内存复用策略，该策略根据每一层内正向传播和反向传播之间的独立性来复用它们的内存空间，即节点的梯度能够重用节点数据的内存空间。这种策略能够使得当前层的内存占用降低50%，适用于层的范围较宽的神经网络。此外作者还观察到：深度神经网络中的计算和内存占用都遵循逐层的方式。因此，对于来自不同层的独立和顺序操作而

言，可以重复地使用相同的内存空间。

Wang等人[22]通过对张量生命周期进行分析，实现了不同的张量在不同的时间复用同一块内存空间的目的。生命周期分析经常在一个训练迭代中动态地保存和释放张量，而一个典型的训练阶段包含数百万个迭代，如果使用cudaMalloc和cudaFree，这种高强度的内存操作会带来巨大的性能开销。为了解决这个问题，作者实现了一个基于堆的GPU内存池，提前分配一个大的GPU内存作为共享内存池。

内存共享一直以来都是优化内存占用的常见思想。通过对数据结构生命周期的分析，可以较好地应用内存共享策略。目前主流的机器学习框架应用了内存共享的策略。

### 3.4 压缩

压缩在深度学习中的应用有多种方式。在深度学习的训练阶段，通过压缩算法对变量进行压缩，能够有效降低变量占据的内存空间，减少加速设备内存的占用；而在深度学习的推理阶段，为了能够将训练好的模型部署在内存受限的边缘设备上，研究者提出了模型剪枝、量化等方法，通过优化模型结构、减少模型参数占用的内存空间等方式对神经网络模型进行压缩。

vDNN的工作能够较好地解决深度神经网络在训练时GPU内存占用高的问题，使得更深层次的神经网络模型能够在单个GPU中进行训练。然而，由于PCIe带宽的限制，当数据移入和移出CPU内存所需的时间比计算DNN的反向传播算法所需的时间长时，vDNN就会产生不可忽略的性能开销。为了解决这个问题，Rhu等人[28]实现了一个压缩直接内存访问（compressing direct-memory-access，cDMA）引擎，它通过降低交换的数据结构的大小来减小PCIe带宽较低带来的性能损失。作者注意到，被广泛用于DNN的ReLU层产生的可交换的数据具有显著的稀疏性和高度可压缩的特征。cDMA引擎利用交换的数据固有的稀疏性，实现了平均提供2.6倍（最大13.8倍）的压缩比，将vDNN的性能平均提高了53%（最高79%）。由于ReLU层被广泛应用于CNN模型，所以cDMA在CNN模型中有较好的性能，在其他类型的神经网络中效果一般。

Jain等人[29]针对训练过程中的特征映射，提出了一种高效的分层编码机制，以减少训练内存的占用。作者注意到在POOL层后的ReLU层的输出结果能够用1 bit的数据来代替之前32 bit的数据，通过这种编码方式能够实现32倍的压缩率；此外CONV层后的ReLU层的输出结果具有高度稀疏性的特征，作者实现了对这种类型的特征映射高效稀疏格式的存储，利用稀疏性减少内存占用。这两种编码方式没有造成对网络模型进行训练时的精度损失。作者对DNN的数据流图进行静态分析，识别出适用的编码方式，并通过插入相关的编码和解码函数创建了一个新的数据流图。

Han等人[30]介绍了“深度压缩方法”，包括模型剪枝、参数量化和哈夫曼编码3个阶段，这3个阶段共同作用，减少了神经网络的存储需求，从而使得神经网络模型能够被部署在内存受限的嵌入式设备中。首先是模型剪枝，作者通过正常的网络训练来学习连接性，修剪小权重的连接：所有权值低于阈值的连接都将从网络中删除；然后，对网络进行再训练，以获取剩下的稀疏连接的最终权值；接着是参数量化，作者通过减少表示每个权值所需的比特数来进一步压缩修剪后的网络，此外作者还通过权值共享让多个连接共享相同的权值，减少了需要存储的有效权值的数量，并且对这些共享的权值进行微调，保证训练精度不受损失；最后，利用哈夫曼编码对网络模型进行进一步的压缩。通过应用“深度压缩方法”，作者

成功将AlexNet的网络模型大小从240 MB降至6.9 MB，VGG16的模型大小从552 MB降至11.3 MB，使得这些神经网络能够部署在智能手机等边缘设备中。

Rhu等人[28]和Jain等人[29]提出的方案都可用于降低DNN在训练过程的内存占用。适用于压缩技术的数据需要具有高度稀疏性的特征，这使得该技术仅能应用于部分数据结构，往往需要与其他技术配合才能发挥较好的效果。而Han等人[30]的工作则致力于降低DNN在部署时的内存占用，使得DNN模型能够被应用到边缘计算设备中。模型剪枝、参数量化等方法会造成模型训练精度的损失，在保持较高精度的同时减小模型占用的内存空间一直以来都是研究的热点。

这些解决方案应用了不同的内存管理技术，各有优点和缺点，具体见**表1**。

## 4 结束语

本文介绍了深度神经网络的一些基本特征，分析了深度学习训练过程中的内存瓶颈，讨论了在深度学习中内存管理面临的挑战。深度神经网络正朝着更深、更广的方向发展，训练和部署这些深度神经网络需要更大的内存空间，这对深度学习系统中的内存管理提出了新的挑战。如何在深度学习系统中进行高效的内存管理，从而满足更深、更广层次的神经网络模型的训练需求，是当前深度学习系统研究的重要问题。为了分析现有的深度学习中内存管理的解决方案，笔者根据其所应用技术的不同对这些解决方案进行分类。内存管理方案主要应用的技术包括内存交换、重计算、内存共享和压缩。通过对现有的一些代表性工作的分析，笔者发现大多数工作通过观察、分析DNN模型训练过程中的一些特征，从数据流图、层以及张量等不同的维度，应用上述的一种或多种技术方案，充分发挥各技术的优势，实现有效的内存管理。从顶层的角度来看，许多工作是类似的，只是它们的实现细节不同。

最后，笔者对在深度学习中进行内存管理的一些新的挑战和机会做如下总结。

表1 应用不同内存管理技术的解决方案对比分析

| 解决方案 | 内存管理技术 | 优点 | 缺点 |
|---|---|---|---|
| vDNN[18] | 内存交换 | 基于分层特征的内存交换策略能够大幅降低GPU内存占用 | 通信和计算不能完全重叠，带来较大的性能开销；适用范围有限 |
| cDMA[28] | 内存交换、压缩 | 在vDNN方案的基础上实现了对交换数据的压缩，能够有效降低性能损失 | 可应用压缩策略的数据类型有限，内存占用降低的效果一般 |
| Layrub[21] | 内存共享、内存交换 | 针对不同类型的神经网络应用不同的内存管理策略，几乎适用于所有的神经网络 | 基于分层策略的内存交换策略会带来一定的性能损失 |
| Gradient-Checkpoint[24] | 内存共享、重计算 | 对于$n$层线性神经网络，该方案的空间复杂度仅为$O\sqrt{n}$ | 通过计算来换取内存，带来额外的性能开销，降低了训练性能；仅适用于线性神经网络 |
| SuperNeurons[22] | 内存共享、内存交换、重计算 | 基于对计算图的静态分析，将3种策略应用于神经网络的训练中，可有效降低训练过程中的内存占用 | 静态分析不能很好地评估不同策略带来的性能损失，此外基于分层的粗粒度的内存管理策略无法实现最优的内存管理方案 |
| Capuchin[23] | 内存交换、重计算 | 基于张量的细粒度的内存管理策略能够对不同的张量应用不同的内存管理策略，充分发挥不同策略的优势，在保证性能的同时，可有效降低内存占用 | 重计算策略带来的性能开销不可忽略，在算力较差的加速设备中性能损失更加明显 |

● 基于虚拟内存实现的内存管理方案。内存交换和内存共享技术都通过构建虚拟内存来扩展加速设备有限的内存资源，提高内存资源的利用率。内存交换技术中交换单元的大小对系统的性能有较大的影响，先前的解决方案以页面为内存交换的基本单位，但是性能较差，现在最好的解决方案是以张量为内存交换的基本单位，在虚拟内存中能够以一个更合适的粒度对内存进行管理，从而实现更好的性能。但是以张量为粒度的方案并不一定是最优的，后续的研究也需要探索更多可能的方案。此外，内存管理策略也十分重要，内存管理策略决定了优化内存占用的效果。目前的研究都朝着这个方向努力，但是还没有很好的内存管理策略能够实现内存占用和计算性能的完美平衡。

● 基于压缩技术实现的内存管理方案。在目前的工作中，压缩技术只能应用于特定的层所产生的数据结构。这些特定的数据结构具有稀疏性和高度可压缩的特征。但是这些特定的数据结构是特定的算法产生的，这为压缩技术的应用带来了很大的限制。随着深度神经网络的发展，研究人员也在不断提出新的算法，试图从数据中提取更多信息，通过新型网络结构的设计提高模型的准确率。未来会有更多新的算法出现，而这些算法产生的数据结构依然可能会具有稀疏性和高度可压缩性的特征，这将为压缩技术的应用带来更多的可能。

● 深度学习编译器优化。深度学习系统的编译器旨在提高系统性能、优化内存使用以及提高模型的可移植性。编译器框架有机会分析和调度内存的使用，消除许多中间变量产生的不必要的内存空间，优化内存使用。这为解决深度学习系统中的内存管理问题提供了新的思路。目前在这方面已经有了一些研究，比如谷歌公司提出的XLA编译框架，用于优化TenosrFlow中计算的子图，可提高计算性能，优化内存占用。但是现有的研究还有很多问题需要解决，距离实现理想的编译器框架构想还有很长的路要走。

## 参考文献:


[1] 陈游旻，李飞，舒继武. 大数据环境下的存储系统构建：挑战、方法和趋势[J]. 大数据，2019, 5(4): 27–40.
CHEN Y M, LI F, SHU J W. Building storage systems in big data era: challenges, methods and trends[J]. Big Data Research, 2019, 5(4): 27–40.

[2] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognition[C]// The 2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE Press, 2016: 770–778.

[3] IOFFE S, SZEGEDY C. Batch normalization: accelerating deep network training by reducing internal covariate shift[C]// The 32nd International Conference on Machine Learning. New York: ACM Press, 2015: 448–456.

[4] KRIZHEVSKY A, SUTSKEVER I, HINTON G. ImageNet classification with deep convolutional neural networks[C]// The 26th Annual Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2012: 1106–1114.

[5] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735–1780.

[6] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[C]// The 3rd International Conference on Learning Representations. [S.l.:s.n.], 2014.

[7] SZEGEDY C, LIU W, JIA Y Q, et al. Going deeper with convolutions[C]// The 2015 IEEE Conference on Computer Vision and

Pattern Recognition. Piscataway: IEEE Press, 2015: 1–9.

[8] ZAGORUYKO S, KOMODAKIS N. Wide residual networks[J]. Computer Science, 2016, arXiv: 1605.07146.

[9] DEVLIN J, CHANG M–W, LEE K, et al. BERT: pretraining of deep bidirectional transformers for language understanding[J]. Computer Science, 2018, arXiv: 1810.04805.

[10] 王孝远, 廖小飞, 刘海坤, 等. 面向大数据的异构内存系统[J]. 大数据, 2018, 4(4): 15–34.
WANG X Y, LIAO X F, LIU H K, et al. Big data oriented hybrid memory systems[J]. Big Data Research, 2018, 4(4): 15–34.

[11] 李鑫, 陈璇, 黄志球. 面向大数据应用的混合内存架构特征分析[J]. 大数据, 2018, 4(3): 61–80.
LI X, CHEN X, HUANG Z Q. Analysis on hybrid memory architecture for big data application[J]. Big Data Research, 2018, 4(3): 61–80.

[12] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient–based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86: 2278–2324.

[13] ABADI M, BARHAM P, CHEN J M, et al. TensorFlow: a system for large–scale machine learning[C]// The 12th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2016: 265–283.

[14] BERGSTRA J, BREULEUX O, FREDERIC B, et al. Theano: a CPU and GPU math compiler in Python[C]// The 9th Python for Scientific Computing Conference. [S.l.:s.n.], 2010: 1–7.

[15] PASZKE A, GROSS S, MASSA F, et al. PyTorch: an imperative style, high–performance deep learning library[C]// The 2019 Annual Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2019: 8024–8035.

[16] CHEN T Q, LI M, LI Y T, et al. MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems[J]. Computer Science, 2015, arXiv: 1512.01274.

[17] DEAN J, CORRADO G, MONGA R, et al. Large scale distributed deep networks[C]// The 26th Annual Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2012: 1232–1240.

[18] RHU M, GIMELSHEIN N, CLEMONS J, et al. vDNN: virtualized deep neural networks for scalable, memory–efficient neural network design[C]// The 49th Annual IEEE/ACM International Symposium on Microarchitecture. Piscataway: IEEE Press, 2016: 1–13.

[19] CHEN M, SUN M M, YANG J, et al. Training deeper models by GPU memory optimization on TensorFlow[C]// Advances in Neural Information Processing Systems 30. [S.l.:s.n.], 2017.

[20] CHEN X M, CHEN D Z, HAN Y H, et al. moDNN: memory optimal deep neural network training on graphics processing units[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(3): 646–661.

[21] JIN H, LIU B, JIANG W B, et al. Layer–centric memory reuse and data migration for extreme–scale deep learning on many–core architectures[J]. ACM Transactions on Architecture and Code Optimization, 2018, 15(3): 1–26.

[22] WANG L N, YE J M, ZHAO Y Y, et al. Superneurons: dynamic GPU memory management for training deep neural networks[C]// The 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York: ACM Press, 2018: 41–53.

[23] PENG X, SHI X H, DAI H L, et al. Capuchin: tensor–based GPU memory management for deep learning[C]// The 24th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 2020: 891–905.

[24] CHEN T Q, XU B, ZHANG C Y, et al.

Training deep nets with sublinear memory cost[J]. Computer Science, 2016, arXiv: 1604.06174.

[25] GRUSLYS A, MUNOS R, DANIHELKA I, et al. Memory-efficient backpropagation through time[C]// The 2016 Annual Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2016: 4125-4133.

[26] KUSUMOTO M, INOUE T, WATANABE G, et al. A graph theoretic framework of recomputation algorithms for memory-efficient backpropagation[C]// The 2019 Annual Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2019: 1161-1170.

[27] JAIN P, JAIN A, NRUSIMHA A, et al. Checkmate: breaking the memory wall with optimal tensor rematerialization[C]// Machine Learning and Systems 2020. [S.l.:s.n.], 2020: 497-511.

[28] RHU M, O'CONNOR M, CHATTERJEE N, et al. Compressing DMA engine: leveraging activation sparsity for training deep neural networks[C]// The IEEE 24th International Symposium on High Performance Computer Architecture. Piscataway: IEEE Press, 2018: 78-91.

[29] JAIN A, PHANISHAYEE A, MARS J, et al. Gist: efficient data encoding for deep neural network training[C]// The 45th ACM/IEEE Annual International Symposium on Computer Architecture. Piscataway: IEEE Press, 2018: 776-789.

[30] HAN S, MAO H Z, DALLY W. Deep compression: compressing deep neural network with pruning, trained quantization and Huffman coding[C]// The 4th International Conference on Learning Representations. [S.l.:s.n.], 2016.

## 作者简介

**马玮良**(1996- ),男,华中科技大学计算机科学与技术学院硕士生,主要研究方向为新体系结构下深度学习系统的优化。

**彭轩**(1995- ),男,华中科技大学计算机科学与技术学院博士生,主要研究方向为分布式深度学习系统平台。

**熊倩**(1997- ),女,华中科技大学计算机科学与技术学院硕士生,主要研究方向为联邦学习。

**石宣化**（1978- ），男，博士，华中科技大学计算机科学与技术学院教授，大数据技术与系统国家地方联合工程研究中心副主任，主要研究方向为并行与分布式计算、多核体系结构与系统软件。当前主要研究云计算与大数据处理、异构并行计算等。

**金海**（1966- ），男，博士，华中科技大学教授，长江学者特聘教授，中国计算机学会会士，IEEE Fellow，ACM终身会员，武汉网络安全战略与发展研究院院长，华中科技大学大数据技术与系统国家地方联合工程研究中心主任，服务计算技术与系统教育部重点实验室主任。主要研究方向为计算机体系结构、计算系统虚拟化、集群计算和云计算、网络安全、对等计算、网络存储与并行I/O等。

**收稿日期：** 2020-05-09
**基金项目：** 国家自然科学基金资助项目（No. 61772218）
**Foundation Item:** The National Natural Science Foundation of China(No. 61772218)