神经网络计算加速方法文献综述

Anonymous

2021-12-05

1 绪论

近期研究表明，在机器学习领域，神经网络（Neural Network, NN）相比其他传统算法具有很大的优越性。例如，在 2012 年，卷积神经网络（CNN）的应用将 ImageNet 数据集上的图像分类准确率前五名，从 73.8% 提升到了 84.7%。其出色的特征提取能力，进一步提高了目标检测能力。而循环神经网络（RNN）则在语音识别领域取得了最高的正确率。[1]

网络模型名称	AlexNet	VGG19	ResNet152	MobileNet	ShuffleNet
年份	2012	2014	2016	2017	2017
参数数量	60M	144M	57M	4.2M	2.36M
运算数量	1.4G	39G	22.6G	1.1G	0.27G
最高准确率	61.0%	74.5%	79.3%	70.6%	67.6%

表 1: 先进的神经网络模型比较

然而神经网络模型的计算和存储复杂度都很高。如上表所示，以 CNN 为例，VGG19 模型用于对 224×224 的图像进行分类，运行一次就需要高达 39 GOP 的算力，仅仅是存储参数就需要超过 500 MB 空间（FP32）。之后的模型虽然尝试降低网络的规模，但牺牲了一定的准确率。[1] 而在模型训练过程中，需要对数据集中的大量数据反复运行模型并进行反向传播，迭代以提高准确率，这对计算资源是巨大的挑战。因此对神经网络的计算进行加速显得很重要。

本文结合 [12]，采用其中的分类方式，将对神经网络的加速方法分为**结构层面**、**算法层面**和**硬件层面**。这种分类方法比较合理，顺序从网络的设计到模型的训练再到硬件实现，涵盖了神经网络的开发全过程。本文假设读者对神经网络有基本了解。

2 结构层面

通过减少网络中的**参数冗余**和**表示冗余**，网络的训练和推断可以得到有效加速。

2.1 参数冗余

2.1.1 层分解（Layer decomposition）

将卷积层中的 $m \times n$ 卷积核权重矩阵 $\mathbf{W}$ 分解为两个满秩矩阵的乘积 $\mathbf{U} \cdot \mathbf{V}$，两个矩阵的大小分别是 $m \times r$ 和 $r \times n$。对于单个卷积层，采用这种低秩近似（Low-rank approximation）可以在加速 2 ~ 2.5 倍的同时，只损失不到 1% 的分类准确率。

而如果把卷积核看作三维的张量，更是可以加速 6 ~ 8 倍；将层分解用于全连接层或网络中的其他模块，如 ReLU 模块等，也能取得 2 ~ 4 倍不等的加速。[12]

2.1.2 网络剪枝（Network pruning）

网络剪枝可以有效减小网络规模，减少过拟合，提高推断速度，并能将网络应用于嵌入式设备等。一种简单的策略是忽略权重小于阈值的连接，在 GPU 上测试发现可以加速 9 ~ 13 倍，而另一种方式可以减少 70% 的神经元。其他剪枝策略包括合并部分神经元，忽略图像中不活跃的通道，也具有一定的加速效果。剪枝与其他加速策略一起使用能取得更好的效果。

然而网络剪枝会引起结构性稀疏（Structured sparsity），对硬件不友好。一些诸如结构化稀疏学习（Structured Sparsity Learning, SSL）等新方法的提出对解决结构性稀疏提供了帮助。[12]

2.1.3 分块循环投影（Block-circulant projection）

一个权重矩阵可以用分块循环矩阵来表示，从而降低网络的存储需求。应用于全连接层或卷积层能取得一定的效果。[12]

2.1.4 知识蒸馏（Knowledge distillation）

知识蒸馏指的是用一个复杂网络输出的合成数据来训练另一个紧凑的小型网络，从而达到对复杂网络的模仿并获得加速效果。如教师—学生网络可以使学生网络具有很高的准确率和较低的复杂度。[12]

2.2 表示冗余

神经网络中的很多权重都很小，所以一般网络中的使用 32 位浮点数来达到较高的准确率。然而浮点数运算较慢，研究发现使用定点数代替浮点数也能取得较好的结果，例如使用 16 位甚至 10 位、8 位的定点数。更加极端的策略是使用 1 位表示，可以细

分为二进制输入、二进制网络和二进制运算。二值化能显著提高速度和降低存储空间要求。此外，随机计算（Stochastic computing）也被引入到神经网络中。然而这些策略对准确率具有一定的负面影响。[12]

3 算法层面

3.1 梯度下降优化

梯度下降算法是将损失函数最小化的最流行的算法。分布式梯度下降能减少硬件负载，提高训练速度，例如 Google 的 CNN 训练实现了两级的并行，一个 CNN 的多份拷贝同时在不同的数据集上训练，并以平均梯度来更新参数，从而反映不同数据集的特征；同一个拷贝的神经元被分配到不同核心上进行计算。在反向传播中加入布谷鸟搜索（Cuckoo search）、蚁群算法（Ant colony algorithm）、遗传算法等非传统算法，能发挥非传统算法的优势。

对于深度神经网络，随着误差累计，梯度可能会衰减到零或增长到越界，引入动量、使用自适应学习率或部分梯度能改进梯度算法。[12]

3.2 卷积优化

有三种方法可以优化卷积计算。基于 im2col 的算法将输入矩阵线性化为多个降低的向量，之后可以高效计算。基于 Winograd 的方法适合计算小型卷积，通过增加加法来减少减法，并降低存储需求。而使用 FFT 算法需要在时域和频域间来回转换，消耗较多的时间和空间，因而只在较大的卷积上具有优化效果。[12]

4 硬件层面

以 FP32 为例，目前先进的 CPU 的运算速度可达几百 GFLOPS，但是还不足以达到高性能计算的要求，而 CPU 的能耗比也较差，使其不能用于对能耗比要求高的移动应用中。尽管在几十年前 CPU 是机器学习的主流硬件，但现在已经很少用 CPU 来实现神经网络。先进的 GPU 可以提供高达几十 TFLOPS 的算力，还有不少使用方便的开发框架，所以 GPU 是高性能神经网络应用的首选硬件。

除了 CPU 和 GPU 这两类通用计算单元，FPGA 具有很高的能源利用率和灵活性，适用于移动应用，但其运行频率较低，很难独立实现高性能计算。[1] 而用于神经网络的 ASIC，知名的主要是张量处理单元（Tensor Processing Unit, TPU），实现了低精度的网络计算，速度可以达到 GPU 的水平，但能源利用率也很高。[12]

4.1 GPU

早在 2004 年，[7] 就利用 GPU 的高度并行性，将神经网络中的卷积和激活函数交给 GPU 着色器（shader）实现，获得了 20 倍的效率提升。

2010 年的一项研究用 CUDA (Compute Unified Device Architecture) 实现了高性能 CNN 库，虽然仅能用于 NVIDIA GPU，但 CUDA C 相比图形 API 更容易理解和学习。CUDA 设备具有数量很多的流处理器（Streaming Multiprocessor），每个流处理器在一个周期内能完成一次融合乘加（FMA），但一个流处理器内的所有线程只能运行同一份代码，这被称为 SIMT (Single Instruction, Multiple Threads)，适合神经网络的计算。在 LeNet5 模型下，与最优的 CPU 实现相比，GPU 实现了 2 ~ 8 倍的加速，且加速比随着图像规模增加而增加。因此 GPU 在处理大网络时具有明显优势。[10]

GPU 也用于多种网络的加速，如 RNN。一项研究实现了 2 ~ 11 倍的加速，其 CPU 实现已经使用了高性能的 Intel Math Kernel Library，能有效使用 CPU 的 SIMD 性能。该模型实现了微软研究所句子补全挑战的 47% 准确率，该挑战的最高准确率为 60.8%。[3]

在多 GPU 联合训练方面也有一些创新。如异步随机梯度下降（Asynchronous stochastic gradient descent, A-SGD）应用于多 GPU，在 8 GPU 上获得了 3.2 倍加速，并观察到 A-SGD 在训练后期效果更明显，而前期可能会扰乱梯度下降的效率，因此可以采用适当的策略来使用 A-SGD。[9]

在一个端到端实时多媒体处理应用中，GPU 相比 CPU 实现了 31.2 倍的加速和 26 倍的能源利用率。[11]

4.2 FPGA 与 ASIC

与软硬件分别设计的 CPU、GPU、DSP 不同，FPGA 只实现算法必要的逻辑，因而能获得更高的效率。FPGA 通常作为神经网络加速器，与 CPU 协同计算，因为 FPGA 自身存储不足以保存庞大的网络。FPGA 的能源利用率可达 10 ~ 100 GOP/J，比 GPU 高很多，尤其是使用低精度计算，如 1 位精度甚至可以接近 1 TOP/J。但是 FPGA 速度不及 GPU，而通过 FPGA 集群会造成能源利用率下降，可能与 GPU 差距不大。[1]

在同一个端到端实时多媒体处理应用中，CPU+FPGA 异构计算相比 CPU 取得了 9.7 倍的加速和 65 倍的能源利用率。[11]

[8] 也展示了 FPGA 的高能源利用率和速度上的相对不足。另一项研究则关注 RNN 的加速，在显示 FPGA 相比 GPU 数倍以上的能源利用率的同时，发现使用 ASIC-64FMA 具有更高的能源利用率，保守估计在相同工艺下也可达 FPGA 的 7 倍。[5] 他们的另一篇文章研究了 1 位精度的 BNN，在多个其他模型上也得到类似的结果。[4]

[6] 则认为新一代的 FPGA 可以超越 GPU，前提也是利用 FPGA 在低精度计算上的优势。文中实现的 2 位精度 ResNet 准确率损失不到 1%，使用 Intel Stratix 10 FPGA 比 Titan X Pascal GPU 效率提高 60%，FPGA 的能源利用率仍有 GPU 的 2.3 倍。

4.3 移动设备

大多数硬件加速研究都关注服务器和桌面平台，但现在移动平台的重要性日渐提高。例如 CNNdroid 实现了 Android 平台下，利用移动设备 GPU 来加速 CNN 推断过程。GPU 实现了 4 ~ 43 倍的加速，与网络和设备类型相关，在复杂网络如 AlexNet 下加速更明显。在能源利用率方面，研究认为 CNNdroid 实现了最高 130 倍的提升。[2]

4.4 新型硬件

IBM 研究人员提出了电阻处理单元（resistive processing unit, RPU），该单元能同时存储和计算参数，能源利用率可达惊人的 84 TOP/J，相当于最先进的 CPU 的 30000 倍。此外，可变电阻存储器（Resistive memories）、忆阻器阵列（Memristor crossbar array）对未来的神经网络加速有广阔前景。[12]

参考文献

[1] Kaiyuan Guo et al. *A Survey of FPGA-Based Neural Network Accelerator*. 2018. arXiv: 1712.08934 [cs.AR].

[2] Seyyed Salar Latifi Oskouei et al. "CNNdroid: GPU-Accelerated Execution of Trained Deep Convolutional Neural Networks on Android". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. Amsterdam, The Netherlands: Association for Computing Machinery, 2016, pp. 1201–1205. ISBN: 9781450336031. DOI: 10.1145/2964284.2973801. URL: https://doi.org/10.1145/2964284.2973801.

[3] Boxun Li et al. "Large scale recurrent neural network on GPU". In: *2014 International Joint Conference on Neural Networks (IJCNN)*. 2014, pp. 4062–4069. DOI: 10.1109/IJCNN.2014.6889433.

[4] Eriko Nurvitadhi et al. "Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC". In: *2016 International Conference on Field Programmable Technology (FPT)*. 2016, pp. 77–84. DOI: 10.1109/FPT.2016.7929192.

[5] Eriko Nurvitadhi et al. "Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC". In: *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. 2016, pp. 1–4. DOI: 10.1109/FPL.2016.7577314.

[6] Eriko Nurvitadhi et al. “Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?” In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA ’17. Monterey, California, USA: Association for Computing Machinery, 2017, pp. 5–14. ISBN: 9781450343541. DOI: 10.1145/3020078.3021740. URL: https://doi.org/10.1145/3020078.3021740.

[7] Kyoung-Su Oh and Keechul Jung. “GPU implementation of neural networks”. In: *Pattern Recognition* 37.6 (2004), pp. 1311–1314. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2004.01.013. URL: https://www.sciencedirect.com/science/article/pii/S0031320304000524.

[8] Kalin Ovtcharov et al. “Accelerating deep convolutional neural networks using specialized hardware”. In: *Microsoft Research Whitepaper* 2.11 (2015), pp. 1–4.

[9] Thomas Paine et al. *GPU Asynchronous Stochastic Gradient Descent to Speed Up Neural Network Training*. 2013. arXiv: 1312.6186 [cs.CV].

[10] Daniel Strigl, Klaus Kofler, and Stefan Podlipnig. “Performance and Scalability of GPU-Based Convolutional Neural Networks”. In: *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*. 2010, pp. 317–324. DOI: 10.1109/PDP.2010.43.

[11] Yuexuan Tu et al. “A Power Efficient Neural Network Implementation on Heterogeneous FPGA and GPU Devices”. In: *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. 2019, pp. 193–199. DOI: 10.1109/IRI.2019.00040.

[12] Qianru Zhang et al. “Recent advances in convolutional neural network acceleration”. In: *Neurocomputing* 323 (2019), pp. 37–51. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.09.038. URL: https://www.sciencedirect.com/science/article/pii/S0925231218311007.