闲话模型压缩之网络剪枝（Network Pruning）篇

原创 ariesjzj 2019-09-15 09:23:26 16604 收藏 92 版权

文章标签： 模型压缩 剪枝 Model compression Pruning AutoML

1. 背景

今天，深度学习已成为机器学习中最主流的分支之一。它的广泛应用不计其数，无需多言。但众所周知深度神经网络（DNN）有个很大的缺点就是计算量太大。这很大程度上阻碍了基于深度学习方法的产品化，尤其是在一些边缘设备上。因为边缘设备大多不是为计算密集任务设计的，如果简单部署上去则功耗、时延等都会成为问题。即使是在服务端，更多的计算也会直接导致成本的增加。人们正在从各个角度试图克服这个问题，如这几年如火如荼的各处神经网络芯片，其思路是对于给定的计算任务用专用硬件加速。而另一个思路是考虑模型中的计算是不是都是必要？如果不是的话，有没有可能简化模型来减少计算量和存储占用。本文主要谈的就是这一类方法，称为模型压缩（Model compression）。它是软件方法，应用成本低，而且与硬件加速方法并不矛盾，可以相互加成。细分来说，模型压缩又可分很多方法，如剪枝（Pruning）、量化（Quantization）、低秩分解（Low-rank factorization）、知识蒸馏（Knowledge distillation）。每一子类方法展开都可以是很大的话题，所以我们一个个来，今天主要只限于pruning方法。

它基于一个假设，或者说目前的共识。就是DNN的过参数化（Over-parameterization）。我们知道，深度神经网络与其它很多机器学习模型一样，可分为训练和推理两个阶段。训练阶段是根据数据学习模型中的参数（对神经网络来说主要是网络中的权重）；推理阶段中将新数据喂进模型，经过计算得出结果。而过参数化是指训练阶段我们需要大量的参数来捕捉数据中的微小信息，而一旦训练完成到了推理阶段，我们并不需要这么多的参数。这样的假设就支持我们可以在部署前对模型进行简化。模型压缩中的pruning和quantization两类方法正是基于这样的前提。模型简化后有很多好处，包括但不限于：1）最直接的好处就是计算量的减小，从而使计算时间更少，功耗更小。2）Memory footprint变小，可以放到更低端的设备上跑。还有个额外的性能好处是本来需要既慢又耗电的DRAM参与，现在有可能放在SRAM就搞定。3）Size更小的包有利于应用发布和更新。如一些手机市场会对应用的大小有限制，另外也有利于车OTA升级。

有了『理论』上的启发后，下一个问题就是how。显然不能拿了模型瞎剪，因为这样精度可以会下降得很厉害以至无法接受。当然，也有情况会在pruning后精度提高的，这说明原模型过似合（overfit）了，pruning起到了regularization的作用。就一般情况下讲，核心问题是成如何有效地裁剪模型且最小化精度的损失。其实这不是一个新的问题，对于神经网络的pruning在上世纪80年代末，90年代初左右就有研究了。如论文《Comparing Biases for Minimal Network Construction with Back-Propagation》提出了magnitude-based的pruning方法，即对网络中每个hidden unit施加与其绝对值相关的weight decay来最小化hidden unit数量。又如上世纪90年代初当时经典的论文《Optimal brain damage》与《Second order derivatives for network pruning: Optimal Brain Surgeon》分别提出OBD和OBS方法，它们基于损失函数相对于权重的二阶导数（对权重向量来说即Hessian矩阵）来衡量网络中权重的重要程度，然后对其进行裁剪。但因为当时的大环境下，神经网络（那时没有deep neural network，只有neural network，或为区分称为shadow neural network）并不是机器学习的一个特别主流的分支，因此之后的很长一段时间也没有大量开枝散叶，但他们对问题的梳理定义和解决问题思路对二十多年后的很多工作产生了深远的影响。到了2012年，我们都知道深度学习一战成名，大放异彩。之后刷榜之风兴起且愈演愈烈，大家的注意力就是提高精度。于是大趋势就是不断地加深加重网络以提高精度，ImageNet准确率每年都创新高。2015-16年期间，Hang Song等人发表了一系列对深度神经网络进行模型压缩的工作。如《Learning both weights and connections for efficient neural networks》，《EIE: Efficient inference engine on compressed deep neural network》。其中《Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding》获得了ICLR 2016的best paper。其中对当时经典网络AlexNet和VGG进行了压缩。结合pruning，quantization和huffman encoding等多种方法，将网络size压缩了几十倍，性能获得成倍的提升。其中对于pruning带来的精度损失，使用了iterative pruning方法进行补偿，可以让精度几乎没有损失，这让

大家意识到DNN参数冗余程度如此之大，可榨油水如此之多。之后这几年，模型压缩领域变得越丰富，越来越多的相关工作衍生出各种玩法。

从network pruning的粒度来说，可以分为结构化剪枝（Structured pruning）和非结构化剪枝（Unstructured pruning）两类。早期的一些方法是基于非结构化的，它裁剪的粒度为单个神经元。如果对kernel进行非结构化剪枝，则得到的kernel是稀疏的，即中间有很多元素为0的矩阵。除非下层的硬件和计算库对其有比较好的支持，pruning后版本很难获得实质的性能提升。稀疏矩阵无法利用现有成熟的BLAS库获得额外性能收益。因此，这几年的研究很多是集中在structured pruning上。Structured pruning又可进一步细分：如可以是channel-wise的，也可以是filter-wise的，还可以是在shape-wise的。

2. 方法介绍

Network pruning的基本思想就是剪裁最不重要的部分。形式化地，可以表达成：

$$\min_{w} L(D;w) + \lambda||w||_0$$

或者将参数裁剪写成约束优化形式：

$$\min_{w} L(D;w) \quad \text{s.t.}||w||_0 \leq \kappa$$

由于L0范数的存在，让该问题成为一个组合优化问题。由于是NP-hard问题，显然不能硬来。

2.1 分类

其中一个最简单的思路就是贪心法，或称为**saliency-based方法。**即按重要性排序，然后将不重要的部分去除。然而问题来了，就是重要性如何衡量。

- 一个最简单的启发就是按参数（或特征输出）绝对值大小来评估重要性，然后用贪心法将那部分干掉，这类称为**magnitude-based weight pruning。**如2016年经典论文《Pruning Filters for Efficient ConvNets》中把权重的绝对值作为衡量其重要性的手段。但训练出来的权重如果不稀疏不利于pruning怎么办，常用的办法是在训练时loss中加regularizer，尤其是L1 regularizer，从而使得权重稀疏化。对于structured pruning来说，我们想获得结构化的稀疏权重，因此常用group LASSO来得到结构化的稀疏权重，如2015-16年的论文《Learning Structured Sparsity in Deep Neural Networks》和《Sparse Convolutional Neural Networks》等。2017年论文《Learning Efficient Convolutional Networks Through Network Slimming》通过一种巧妙的方法，基于BN（Batch Normalization）层的广泛使用，在BN层加入channel-wise scaling factor 并对之加L1 regularizer使之稀疏，然后裁剪scaling factor值小的部分对应权重。另外，重要性评估也可以针对activation（激活值，即激活函数的输出），当然就算是针对activation最后也会体现在权重的裁剪上。一般来说，像Relu这样的激活函数会倾向产生稀疏的activation；而权重相对而言不太容易是稀疏的（当前，如前所说，我们可以通过regularizer这种外力使它变得稀疏）。从这种意义上说，activation更适合pruning。如2016年论文《Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures》中提出采用Average Percentage of Zeros，即APoZ来衡量activation的重要性。它定义为activation中为0的比例。

- 上面方法有个假设就是参数绝对值越小，其对最终结果影响越小。我们称之为"'smaller-norm-less-important"准则。然而这个假设未必成立（如2018年的论文《Rethinking the Smaller-Norm-Less-Informative Assumption in Channel Pruning of Convolution Layers》有对其的讨论）。第二种思路就是考虑**参数裁剪对loss的影响。**其实前面提到的始祖级的OBD和OBS就是属于此类。但这两种方法需要计算Hessian矩阵或其近似比较费时。近年来有一些基于该思路的方法被研究和提出。如2016年论文《Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning》也是基于Taylor expansion，但采用的是目标函数相对于activation的展开式中一阶项的绝对值作为pruning的criteria。这样就避免了二阶项（即Hessian矩阵）的计算。2018年论文《SNIP: Single-shot Network Pruning based on Connection Sensitivity》将归一化的目标函数相对于参数的导数绝对值作为重要性的衡量指标。

- 第三个思路是考虑对**特征输出的可重建性**的影响，即最小化裁剪后网络对于特征输出的重建误差。它的intuition是如果对当前层进行裁剪，然后如果它对后面输出还没啥影响，那说明裁掉的是不太重要的信息。典型的如2017年论文《ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression》和《Channel pruning for accelerating very deep neural networks》都是通过最小化特征重建误差（Feature reconstruction error）来确定哪些channel需要裁剪。前者采用贪心法，后者用的LASSO regression。2017年论文《NISP: Pruning Networks using Neuron Importance Score Propagation》提出只考虑后面一两层啥的不够，于是提出NISP（Neuron importance score propagation）算法通过最小化分类网络倒数第二层的重建误差，并将重要性信息反向传播到前面以决定哪些channel需要裁剪。2018年论文《Discrimination-aware Channel Pruning for Deep Neural Networks》提出一种比较有意思的变体。它提出DCP（Discrimination-aware channel pruning）方法一方面在中间层添加额外的discrimination-aware loss（用以强化中间层的判别能力），另一方面也考虑特征重建误差的loss，综合两方面loss对于参数的梯度信息，决定哪些为需要被裁剪的channel。

- 另外还有基于其它的准则对权重进行重要性排序。如2018年的论文《Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration》讨论了magnitude-based方法的前提与局限（即需要其范数值方差大，且最小值接近于0）。它提出了 FPGM（Filter Pruning via Geometric Median）方法。其基本思想是基于geometric median来去除冗余的参数。

我们知道贪心算法的缺点就是只能找到局部最优解，因为它忽略了参数间的相互关系。那自然肯定会有一些方法会尝试**考虑参数间的相互关系，试图找导全局更优解。**

- **离散空间下的搜索**：如2015年的论文《Structured Pruning of Deep Convolutional Neural Networks》基于genetic algorithm与particle filter来进行网络的pruning。2017年的论文《N2N Learning: Network to Network Compression via Policy Gradient Reinforcement Learning》尝试将网络的压缩分成两个阶段-layer removal和layer shrinkage，并利用强化学习（Reinforcement learning）分别得到两个阶段的策略。
- **规划问题**：如比较新的2019年论文《Collaborative Channel Pruning for Deep Networks》提出CCP（Collaborative channel pruning）方法，它考虑了channel间的依赖关系，将channel选取问题形式化为约束下的二次规划问题，再用SQP（Sequential quadratic programming）求解。
- **Bayesian方法**：如2017年论文《Variational Dropout Sparsifies Deep Neural Networks》提出了sparse variational droput。它对variational droput进行扩展使之可以对dropout rate进行调节，最终得到稀疏解从而起到裁剪模型的效果。
- **基于梯度的方法**：回顾上面问题定义中的数学最优化问题，其最恶心的地方在于regularizer中那个L0-norm，使目标不可微，从而无法用基于梯度的方法来求解。如2017年的论文《Learning Sparse Neural Networks through L0 Regularization》的思路是用一个连续的分布结合 hard-sigmoid recification去近似它，从而使目标函数平滑，这样便可以用基于梯度的方法求解。
- **基于聚类的方法**：一般地，对于压缩问题有一种方法就是采用聚类。如将图片中的颜色进行聚类，就可以减小其编码长度。类似地，在模型压缩中也可以用聚类的思想。如2018年的论文《SCSP: Spectral Clustering Filter Pruning with Soft Self-adaption Manners》和《Exploring Linear Relationship in Feature Map Subspace for ConvNets Compression》分别用谱聚类和子空间聚类发掘filter和feature map中的相关信息，从而对参数进行简化压缩。

2.2 Sparsity Ratio

上面的很多方法，研究的是给定裁剪量的前提下如何做pruning，如采用什么样的criteria去做参数选取。然而其实还有个核心问题是在哪里裁剪多少，即sparsity ratio的确定。这里的sparsity ratio定义为层中为0参数所占比例，有些文章中也称为pruning rate等。从目标结构或者sparsity ratio的指定方式来说，按2018年论文《Rethinking the Value of Network Pruning》中的说法可分为预定义（**predifined**）和自动（**automatic**）两种方式。Predefined方法由人工指定每一层的比例进行裁剪，因

automatic方法会根据所有的layer信息（即全局信息）由pruning算法确定每层裁剪比例，因此目标结构一开始并不确定。

- 从某种意义上来说，著名的MobileNet《MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications》中提出的multiplier参数也是一种predefined的network pruning方法。只是它比较简单粗暴，将相同的裁剪比例用于所有层，且裁剪后权重不重用。2019年在《EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks》中提出的EfficientNet将这种参数调节更进一步，提出compound scaling method将width, depth, resolution按特定比例一起调节。但它们调节参数都是针对网络中所有层的，粒度比较粗。显然，网络中不同层对于pruning的敏感（sensitivity）程度是不一样的，只有根据层的属性为每层设置最适合的sparsity ratio才是最优的，这种为每层专设的称为local sparsity，相对地前面那种就称为global sparsity。2015和2016年经典论文《Learning both Weights and Connections for Efficient Neural Networks》和《Pruning Filters for Efficient ConvNets》中对网络中各层的敏感度进行了分析，发现卷积层比全连接层对pruning更敏感（直觉上，由于卷积层中的参数共享，所以参数量会比全连接层少得多），另外不同的卷积层之间敏感度也不一样，如第一层卷积对pruning就相对更敏感，ResNet中每个stage中第一个和最后一个residual block比中间block更敏感等。这里就引申出一个重要问题，就是如何确定每一层的最优的sparsity ratio。在predefined这类方式中，一种就是通过前面的提到的sensitivity analysis进而确定每一层的sparsity ratio。
- 另观automatic方法，如上所说，它需要pruning算法从全局信息中自动得到每层裁剪多少。上面提到过的论文《Learning Efficient Convolutional Networks through Network Slimming》就属于此类。另外，2019年论文《Play and Prune: Adaptive Filter Pruning for Deep Model Compression》将pruning问题建模成min-max优化问题，然后通过两个模块交替迭代分别进行裁剪和通过调节pruning rate控制精度损失。2018年论文《ADC: Automated Deep Compression and Acceleration with Reinforcement Learning》提出ADC（Automated deep compression）方法，根据不同需求（如保证精度还是限制计算量），利用强化学习来学习每一层最优的sparsity ratio。2018年论文《“Learning-Compression” Algorithms for Neural Net Pruning》提出Learning和Compression两步交替优化的pruning方法，在Compression操作中，通过将原参数向约束表示的可行集投影来自动找到每层的最优sparsity ratio。因为此类方法不需要计算量较大的sensitivity analysis，也减少了超参数的引入。因此，近年来，这类方法的研究越来越多。

2.3 精度恢复

当模型经过pruning，一般会带来精度损失，因此我们在pruning的同时也需要考虑精度的恢复：

- 前面提到的论文《Channel Pruning for Accelerating Very Deep Neural Networks》中在进行channel pruning后，直接通过least square来得到最小化特征重建精度下的新参数，因此不需要fine-tuning来恢复精度，是一种**inference-time pruning**方法。它的好处是pruning过程不需要训练环境。
- One-shot pruning指一趟头的裁剪，但这种往往对精度影响很大。人们发现裁完后进行fine-tuning可以弥补pruning带来的精度损失，因此很多方法会在pruning后做fine-tuning。比较经典的是training，pruning，fine-tuning三段式。后面两个阶段交替进行，每次pruning后损失的精度可以由后面的fine-tuning来弥补，该过程也称为**iterative pruning**。简单说就是砍一刀回点血，再砍一刀再回点血这样，不一步到位是因为有些实验表明一下子砍太狠就难回血了。当然现实中可以衍生出很多玩法。而在时间维度上，每步砍多少也是有艺术的。如2017年论文《To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression》提出一种automated gradual pruning算法，它基于开始阶段冗余多可以裁得快，越到后面越裁得快的指导思想，给出在n步的pruning中，如何从初始sparsity ratio渐变到目标sparsity ratio的方法。

2.4 重新审视假设

比较有意思的是，最近不少工作开始反思一些之前固有的假设。比如一开始提到的**over-parameterization对训练是否真的那么有益**，还有**原网络的权重是否在pruning中很重要**。在ICLR2019的best paper《The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks》提出The Lottery Ticket Hypothesis，即一个随机初始化，密集的网络包含一个子网络，这个子网络如果沿用原网络的权重初始化，在至多同样迭代次数训练后就可以比肩原网络的测试精度。同时它还给出了找这种子网络结构的方法。文章认为这个子结构和它的初始值对训练的有效性至关重要，它们被称为『winning logttery tickets』。另一篇论文2018年的《Rethinking the Value of Network Pruning》提出不仅over-parameterization对于训练不是重要的，而且从原网络中重用其权重也未必是很好的选择，它可能会使裁剪后的模型陷入局部最小。如果原网络的权重或其初始值不重要的话，那剩下最重要的就是pruning后的网络结构了。换句话说，某种意义上来说，pruning即是neural architecture search（NAS），只是由于 它只涉及层的维度，搜索空间相比小一些。但这也是它的优点，搜索空间小了自然搜索就高效了。

2.5 保留模型Capacity

Pruning按最初的字面意思理解就是给模型做减法。之前的主流pruning方法中，被裁剪的部分一般也就直接丢弃不会再拿回来了，即模型的capacity在iterative pruning的过程中不断减少。这样的话，一旦有参数被不适当地裁剪掉，便无法被恢复。而这两年，学界正在尝试**在模型压缩过程中保留被裁剪部分能力或者扩充能力的方法**。2018年论文《Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks》提出SFP（Soft filter pruning）让被裁剪的filter在训练中仍能被更新，这样它仍有机会被恢复回来。2016年论文《Dynamic Network Surgery for Efficient DNNs》在pruning的基础上加了splicing操作，避免不合适的pruning带来的影响。2017年的论文《Morphnet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks》也是基于这种思路，它会迭代地进行shrink和expand的操作。

Pruning的副作用就是可能会损害模型的capacity。尽管前面的各种方法让该影响尽可能小，但我们往往只能在有限的数据集上做。因此，很可能对于大部分简单的样本pruning对其没有影响，但对于小部分难的数据会有较大影响。那有没有可能在保留网络capacity的同时又能精简计算呢？一些学者尝试**结合dynamic NN**来达到该目的，即网络执行哪些模块由输出决定。如2017年论文《Dynamic Deep Neural Networks: Optimizing Accuracy-Efficiency Trade-offs by Selective Execution》引入Dynamic Deep Neural Networks，对于给定输入，哪部分神经元被执行由网络本身中的controller module来确定。这个module通过强化学习进行训练。另一篇2017年论文《Runtime Neural Pruning》将pruning建模为马尔可夫决策过程（Markov decision process）并通过强化学习来学习pruning策略。当然，这类方法也有些局限，如由于保留了网络原始capacity，因此size不会减少。另外由于执行哪部分是动态的，因此对于硬件加速会有影响（如算子间的fusion）。

3. 小结

和DNN训练不同，DNN的部署有个很大的挑战是平台的多样性。云和端上的硬件平台差异很大，端设备与端设备之间在硬件种类和能力上差异也很大。对于加速硬件的多样性问题，业界已有诸多研究和尝试，如像基于编译器的TVM，和runtime+厂商backend的Android NN runtime。但计算能力的差异仍然是个大问题。如同样是手机，计算能力可能有巨大的差异。这本质上是一个考虑性能与精度等指标的多目标优化问题，我们需要针对不同平台在其pareto解集上找合适的trade-off 。如果要将同一DNN适用于多种平台，需要按不同算力进行裁剪并且训练，这个成本很大（几乎与平台种类线性相关），可伸缩性差。而模型压缩成为了解决多类型平台统一部署方案的希望之一。像2018年论文《Slimmable Neural Networks》考虑不同移动端平台训练同一网络，对于不同计算能力的平台进行不同程度的裁剪。2019年论文《Once for All: Train One Network and Specialize it for Efficient Deployment》也是为了类似的目标，将一个大网络通过progressive shrinking approach训练不同超参下的子网络，在部署时根据不同的精度要求和资源约束搜索合适的子网络。即一次训练，可以适应多种平台部署，因此称为Once for All（OFA）方案。它既可以说是一种模型压缩方法，也可以说是一种神经网络结构搜索（Neural archtecture search）。NAS相关介绍可参见之前写的杂文神经网络架构搜索（Neural Architecture Search）杂谈。类似的，2018年论文《ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware》中其实是用pruning的方法来做NAS。

可以看到，这几年network pruning作为模型压缩中的主力之一，正在受到越来越多的关注。当然，各种更好的pruning参数选取方法一定还会层出不穷。另外，从整体来看，以下几个方向值得关注：

- 如前面提到，network pruning方法与NAS的界限已经模糊了。事实上，NAS分支上也有一类搜索加速方法，如One-Shot Architecture Search是先有一个大网络，然后做减法。NAS与模型压缩两个一开始看似关系不是那么大的分支最后似乎走到一块去了。这两个分支今天有了更多的交集，也必将擦出更多的火花。
- 挑战已有的固有的假设。如前面对于over-parameterization与重用已有参数是否有有益的反思非常有意思。这样的工作会给我们非常大的启发，从而根本改变解决问题的思路。
- 随着AutoML的大潮，越来越多的东西开始自动化。模型压缩能拉下吗？当然不能。经过前面的介绍我们知道，像ADC，RNP，N2N Learning这些工作都是试图将pruning中部分工作自动化。而且对于其它的模型压缩方法，如quantization，也有一些空间可以自动化，如2018年论文《HAQ: Hardware-Aware Automated Quantization》考虑网络中不同层信息的冗余程度不一样，因此可以用不同位数进行量化。
- 这几年机器学习最火热的分支之一GAN，正在不断渗透到已有领域，在pruning中也开始有它的身影。如2019年论文《Towards Optimal Structured CNN Pruning via Generative Adversarial Learning》采用了GAN的思想，让generator生成裁剪后网络，discrimintor来判别是否属于原网络还是裁剪后网络，从而进行更有效的网络结构化裁剪。
- 与硬件结合，如稀疏计算的支持。现在虽然有像cuSPARSE这样的计算库，但底层硬件GPU本身设计并不是专门为稀疏数据处理打造的。如果能将稀疏计算和处理能力做进芯片那必将极大提高计算效率，如早些年有像EIE这样的尝试。在现在神经网络芯片的大潮下，相信这样的案例会越来越多。
- 如文章一开始提到的，模型压缩方法中pruning只是其中一种，它与其它方法并不冲突，因此与其它方法，如knowledge distillation，quantization等的深度结合，是值得研究的方向。
- 和其它机器学习分支一样，很多人提出很多算法，各家都说自家的好。一个分支发展到一定时候，就需要benchmark来进行客观的横向比较。Google在2019年论文《The State of Sparsity in Deep Neural Networks》正是这方面的尝试。相信以后也会有越来越多的benchmark，和针对性的竞赛。