深度神经网络加速技术的剪枝与量化研究

原模型

- 模型复杂，训练困难
- 带宽要求高
- 冗余计算多
- 嵌入式设备难以部署

模型压缩思路

- 模型结构
    1. 设计新组件（Dropout层）
    2. 网络结构搜索——从预定义的大搜索空间中搜索高效的网络结构
    3. 知识蒸馏——从复杂的教师模型中学到的知识生成一个简单的学生模型
- 网络优化
    1. 计算卷积优化——将全连接运算变为卷积运算就是一种卷积优化，减少了参数计算量

       快速傅里叶变换（FFT）、图像到列
    2. 参数分解：将较大的层分解成许多较小的层，从而减少了计算的数量，常用于剪枝和量化
    3. 剪枝：删除不影响网络精度的冗余参数
    4. 量化：通过减少数据的表示精度来减少模型的计算量
- 硬件加速：底层加速库BLAS。

卷积神经网络

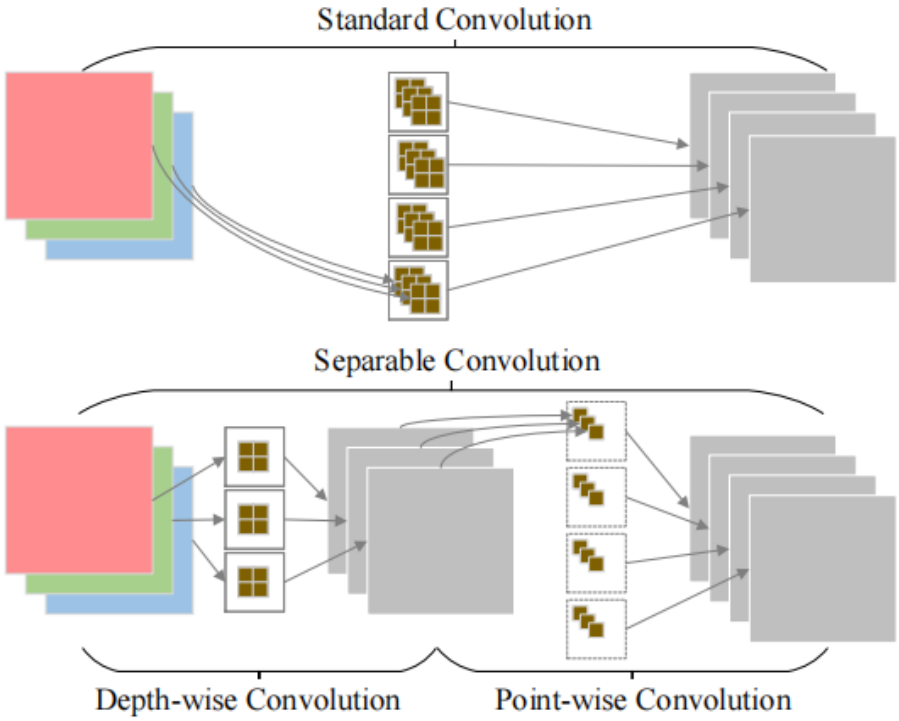
1. 微调被定义为对以前训练过的模型进行再训练；与从零开始训练相比，使用微调更容易恢复量化后的或修剪后的模型的准确性。
2. 未压缩的比较庞大的网络：由于内存、带宽或处理方面的限制，在移动设备上执行的推理时，网络的大小可能会受到限制
3. 本文着重研究了CNN推理分类的加速问题。并在ImageNet、CIFAR和MNIST等测试集上比较不同的模型压缩技术
4. **卷积操作**：

$$\mathbf{F}_n^{l+1} = \mathbf{A}_n^l = \text{activate}\left\{\sum_{m=1}^{M}\left(\mathbf{W}_{mn}^l * \mathbf{F}_m^l\right) + \mathbf{b}_n^l\right\}$$

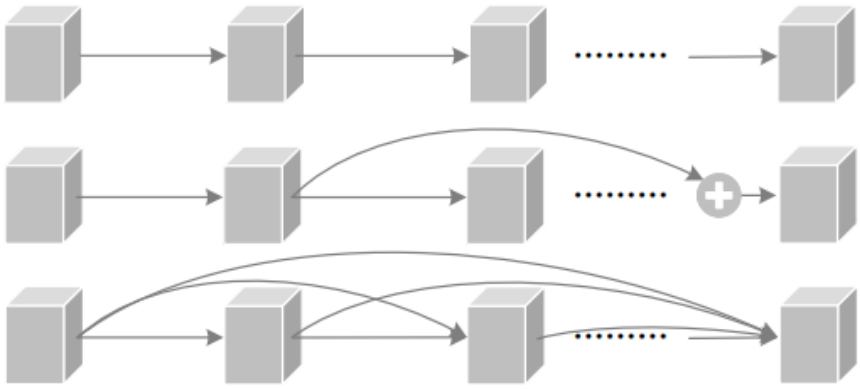
   压缩的目标是在不影响精度的情况下减少权重和输入特征的大小

5. **高效的网络结构**：
    - 可分离的卷积：
        - 深向卷积：将输入特征的每个通道进行分组，将每个通道作为单个输入张量，产生有相同信道数量的激活

        - 点向卷积：点卷积是1×1核的标准卷积。以最小的计算复杂度提取交互信息



    - 卷积块：
        - 普通连接：
        - 残差连接：短路连接，保持前一个块的信息，防止梯度消失，可以增加网络的深度，提高网络表达能力
        - 紧密连接：将前面一个块的信息传递到后面所有的块从而扩展了网络的深度和**感受野**
            - 感受野：（CNN每一层输出的特征图上的像素点在原始图像上的映射的区域大小）



剪枝

修剪已经训练过的网络，修剪之后的网络不需要重新训练，删除了对结果的准确性没有显著贡献的冗余参数或神经元

冗余指的是权重系数为0或者接近0或者为重复值

修剪后的网络被重新训练，防止之前的网络陷入局部极小值，进一步提高准确性

- 剪枝方法三种分类方式：
    1. 结构化与非结构化
    2. 神经元修剪和连接修剪取决于修剪的元素类型（修剪激活值/权重）

3. 静态剪枝和动态剪枝

   静态剪枝是指在推理之前脱机执行所有剪枝步骤，而动态剪枝是在运行时执行的。

- 修剪可以逐元素、逐行、逐列、逐过滤或逐层进行。通常，逐元素的剪枝对模型稀疏性影响最小，并导致一个非结构化模型

$$\arg\min_{p} \boldsymbol{L} = N(x;\mathbf{W}) - N_p(x;\mathbf{W}_p)$$

$$\text{where } N_p(x;\mathbf{W}_p) = \boldsymbol{P}\left(N\left(x;\mathbf{W}\right)\right)$$

- **静态剪枝**：在训练之后和推理之前将神经元从网络中去除，推理时不需要额外修剪，三个部分：
  1. 选择需要修剪的参数
  2. 修剪神经元的方法
  3. 修剪之后的网络微调或者重新训练
  - 剪枝标准：

    两种可能：0值的权重和特征导致0输出；相似的权重和特征可以合并

  - 剪枝方法：
    - **基于幅度的剪枝**：较大的权重比较小的权重更重要，识别不被需要的权重以便在评估过程中被删除；最直观的基于幅度的剪枝方法是修剪所有零值权重或在绝对值阈值内的所有权值。
      1. OBD、OBS——**衡量权重**，这些早期的方法基于损失函数对权重的二阶导数减少了连接的数量。
      2. APoZ：**衡量激活值**的方法，ReLU使得神经元的激活值以很大概率为0，从而易于修剪
    - **基于惩罚项的剪枝**，目标是在训练过程中修改一个错误函数或添加其他约束条件，称为偏差项。偏差项使权重更新为0或接近0，随后进行修剪；LASSO、组LASSO
    - **修剪与微调或再训练相结合**：

      早期方法：剪枝消除了网络冗余，减少了计算的数量，而对某些网络架构的精度没有显著影响。一些重要的因素可能会被消除，从而导致准确性的下降，可以通过**微调或者再训练提高准确性。**

      - **可恢复的修剪**：使修剪后的网络恢复到修剪前的状态，修剪后的元素也可以参与到后续的训练过程中，并进行调整以适应修剪后的网络；通常使用**二进值掩码矩阵**来指明单个权值是否被修剪。l_1范数修剪后的权值可以随机拼接回网络中——导致非结构化的网络，使硬件实现复杂化；**软过滤器修剪**（SFP）进一步扩展了可恢复的修剪。SFP获得了结构化压缩结果，增加了额外的好处或减少了推理时间
      - **增加稀疏性**：应用微调的另一个动机是为了增加网络的稀疏性
      - **自适应稀疏性**：分层修剪比率通常需要人工决定。过高的比率导致非常高的稀疏性可能导致网络发散需要重调；**Network slimming**——网络瘦身，通过自动计算分层稀疏性来实现每层自适应修剪比率；剪枝也可以使用**最小-最大优化模块**来执行，该模块通过保持剪枝比来在调优期间保持网络的准确性。**自动剪枝**将三阶段管道的剪枝和微调作为一个独立的训练层。该层在训练过程中帮助逐渐修剪，最终形成一个相对精简的网络
      - **再训练过程**

        彩票：舍弃原始权重，再训练过程中进行权重初始化，在相同的训练迭代次数下，可以有效地训练密集的、随机初始化的修剪子网络，并达到与原始网络相当的精度。

- **动态剪枝**：静态剪枝无法恢复已被修剪的网络的信息；动态修剪决定了在运行时哪些层、通道或神经元不会参与进一步的运算

  缺点：在此过程中会增加了带宽和计算消耗，如何在动态剪枝过程中平衡资源消耗和网络精度损失是很重要的问题，一个解决方式是抑制PE（加速器中的运算单元）内0值参数计算

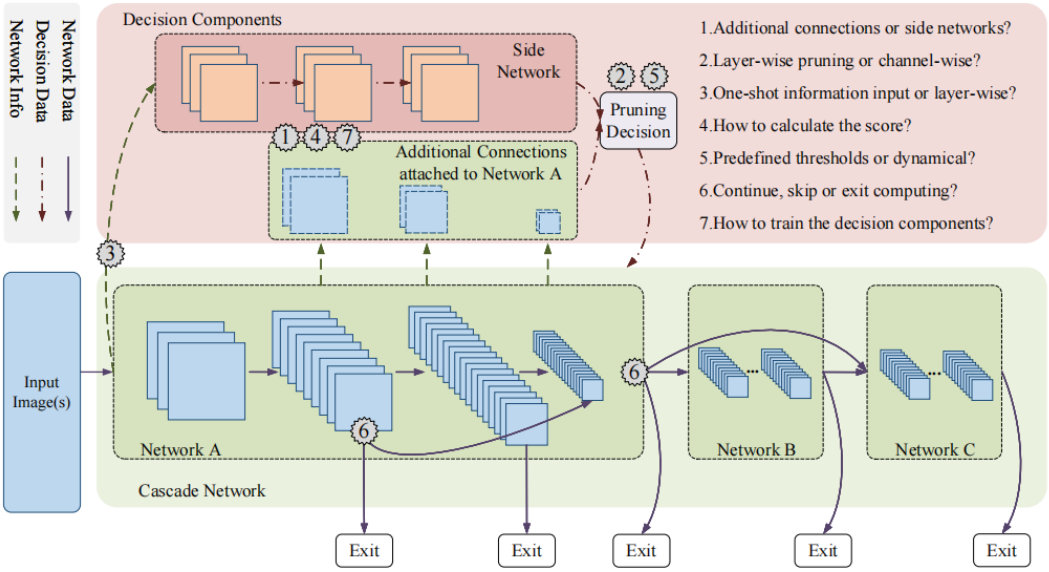


Figure 11: Dynamic Pruning System Considerations

  - 修剪标准：激活网络中的最优部分，不被激活的神经元被修剪
  - 强化学习自适应网络：通过阈值来平衡网络精度损失与计算消耗，自适应网络有多个中间分类器使得网络可以尽早结束推理
    - 级联网络：
      - 级联神经网络（Cascade Neural Network）是一种多层神经网络，每一层都是独立的小网络，用于对输入数据进行预处理和特征提取，级联网络是串行网络的组合，每个网络都有输出层，而不是只有一个输出；因此级联神经网络可以实现早期退出，不需要计算所有的输出层；**缺点**是需要人为设置超参数；
      - 级联网络是一个自适应的网络，带有一个经RL训练的合成器，它可以为每一个输入确定一个合理的计算图；自适应控制器"策略偏好"用于智能地增强合成器，允许从子图中调整网络计算图。
    - 策略网络

      使用RL进行特定图像的训练，以确定剩余网络块是否应该参与之后的计算。只在加载图像时运行一次，生成一个布尔向量来表示哪些残余块被激活，哪些不被激活。
    - RNP（运行时神经修剪）

      一个动态修剪神经网络的框架，它将特征选择问题定义为马尔可夫决策过程，通过强化学习训练一个基于RNN的决策网络，不是移动某些层，RNP预测哪些层是不被需要的
  - 可微的自适应网络
    - 特征增强或抑制（FBS）

      动态选择要跳过或处理哪个通道的方法，是一个引导信道放大和遗漏的侧网络，它使用损失函数被L1约束的SGD算法和卷积网络进行训练
    - T加权决策

$$h(x) = \begin{cases} \sigma(x), & \text{if } x > T \\ 0, & \text{otherwise} \end{cases}$$

删除分数低于给定阈值的通道，通过sigmoid函数计算分数，通过迭代训练寻找阈值

- **组合系统**

  包含一些在图像处理方面表现较好的网络如AlexNet, ResNet, GoogLeNet。将这些网络作为预训练组件格式化为有向无环图，通过评估该图来确定退出的叶子结点

- **多尺度密集网络**（MSDNets）——一种自适应网络；

  1)随时预测，可以在许多节点上生成预测结果，以促进网络的早期退出

  2)批处理计算预算，以强制执行一个更简单的退出标准，如计算限制。结合多尺度的特征图和密集的连通性，来实现准确的早期退出

- 解决自适应网络的训练复杂性：
  1. 梯度平衡：跨多个不同的网络层使用多个中间分类器来帮助主干网络收敛，改善了在MSDNets中发现的梯度不平衡问题
  2. 内联子网协作（ISC）和一对一的知识蒸馏（OFA）：ISC将早期的预测引入到以后的预测器中，以增强它们的输入信息；OFA使用一个最终的分类器来监督所有的中间出口。
  3. 柔性神经网络（SNN）是一种可以在不同宽度下执行的网络。也被称为可切换网络，该网络能够动态地选择网络架构（宽度），而不需要太多的计算开销。

- **不同剪枝方法的比较**
  - Shrinkbench：收缩式工作台，一个统一的基准测试框架，提供剪枝性能的比较
  - Liu：《 Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm.》提出剪枝后的模型可以通过权重初始化+再训练之后恢复模型精度，模型性能只与剪枝后的模型架构有关，与NAS类似
  - the lottery ticket：彩票，认为剪枝后的模型进行再训练时，模型初始化只有与未剪枝时的参数相同时才能恢复精度
  - Glae：《The State of Sparsity in Deep Neural Networks. 》提出非结构化的剪枝方法得到的模型通过微调恢复精度，结构化的剪枝方法得到的模型通过再训练恢复精度；通过比较dropout和L1正则化，得出基于幅度的剪枝更有效

量化

量化指用一组连续的符号和整数值逼近一个连续的信号的过程，类似于聚类和参数共享的概念；

将浮点参数转换为低数值精度的量化神经网络，通过降低精度值来压缩CNN

大部分复杂的神经网络都采用32位浮点数进行表示，而现实往往不需要如此高的精确度，进行低位表示能够在满足精度要求的情况下，减少带宽和存储开销

post train quantize（PTQ）：训练后量化，对原模型微调/对量化模型再训练，在此过程中，真实权值被缩小位整数值，通常为8位表示

quantzation-aware（QAT）：量化感知训练，边训练边量化；前向进行量化和反量化，后向梯度回传为浮点数

饱和量化是指用一种带有校准集的校准算法生成特征尺度，量化后的激活值与量化前的实值有相似的分布——KL散度判断两分布的差异程度

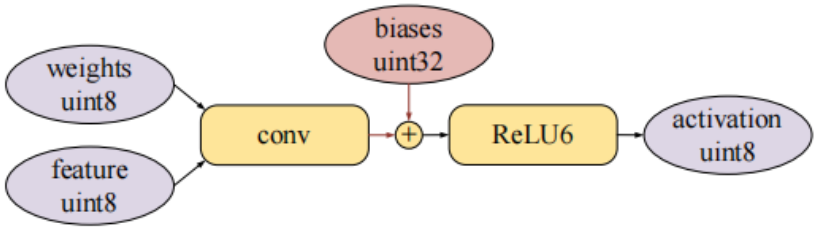
KL散度：kl散度是表示两组之间概率分布相对熵的度量，相对熵，信息散度，P为原始数据，Q为量化之后的数据；用于衡量两个分布之间的差异程度

$$D_{\mathrm{KL}}(P\|Q) = \sum_{i=0}^{N} P(x_i) \log \left(\frac{P(x_i)}{Q(x_i)} \right)$$

量化研究**两个重点**：量化感知训练（QAT）和训练后量化（PTQ）

根据数据分组进行分类：layer-wise和channel-wise，在评估参数宽度时，可以根据长度进一步分类：n位量化。

- **量化方法**
  - 低位宽量化
    - 推理时将FP32->8-bit integers无精度损失



  - 对数量化

    又称为移位量化，将数值从浮点数的形式量化为一个整数或定点数，但它与线性量化不同，两个相邻数之间是在以2为底的对数域上均匀分布的，这使得实际推理当中可以直接通过移位运算来快速实现，同时也拥有随比特数增长而指数增长的大动态范围。
    移位量化既可以只量化权重（对激活值移位），也可以只量化激活（对权重值移位），当然也可以同时量化权重和激活（对值1移位）

  - 正负量化——（-1，+1）
    - 将所有权重表示成1位，类似于对数量化
    - 权重和激活值都表示成1位，0和1分别被编码为表示-1和+1
    - 卷积操作被分离成乘法和加法操作

$$\boldsymbol{x} \cdot \boldsymbol{y} = \mathrm{bitcount}(\mathrm{and}(\boldsymbol{x}, \boldsymbol{y})), \ \text{s.t.} \ \forall i, x_i, y_i \in \{0, 1\} \quad (23)$$

$$\mathrm{x} \cdot \mathrm{y} = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} 2^{m+k} \, \mathrm{bitcount} \left[\mathrm{and} \left(c_m(\mathrm{x}), c_k(\mathrm{y}) \right) \right], \quad \text{s.t.} \ c_m(\mathrm{x})_i, c_k(\mathrm{y})_i \in \{0, 1\} \forall i, m, k. \quad (24)$$

      - 消除复杂的浮点乘法，可以使用简单的累加硬件实现加速计算
      - 二值化将网络规模减少了多达32×，减少了内存的使用，降低了资源消耗
      - 信息损失严重，精度降低，相比于32位浮点精度，网络表现欠佳
    - **两种转换方法**：随机性二值化、确定性二值化

      随机方法考虑全局统计量或输入数据的值来确定某个参数为-1或+1的**概率**；确定性二值化直接根据一个阈值计算位值（阈值通常是0），从而得到一个符号函数。确定性二值化在硬件中实现要相对简单。

    - 二元连接：BC是由库巴里奥提出的，是一种早期的神经网络二元化的随机方法，将正向和反向传播中的权值进行二值化；在小数据集上表现较好，不适用于大数据集

$$x^b = \begin{cases} +1, & \text{with probability } p = \sigma(x) \\ -1, & \text{with probability } 1 - p \end{cases} \tag{25}$$
$$\text{where } \sigma(x) = \text{clamp}\left(\frac{x+1}{2}, 0, 1\right)$$

- 库巴里奥将二值化应用于激活函数，形成了第一个BNN——二值化神经网络
- 拉斯特加里在此基础上提出了BWN——二进制权值网络，引入了一个缩放系数α，使得$W = \alpha B$，并将二值化应用到XNOR-NET
- DoReFa-Net：也应用了正负算法，同时在反向传播时将8位梯度量化位低位表示；梯度在反向传播中被随机量化。例如，它需要1位来表示分层的权重，2位的激活，以及6位来表示梯度。
- Ternary Weight Networks (TWN)——三元权重网络：将权重表示为三元值-1，0，1；与BNN相比保留了更多的信息同时保持了较高的计算效率
- Ternary Neural Networks (TNN) ——三元神经网络：权重激活值都表示为三元；从教师神经网络中学到学生模型（输出激活值为三元）——知识蒸馏
- Fine-Grained Quantization (FGQ)：英特尔提出细粒度量化
- MeliusNet是一个由两种类型的二进制块组成的二进制神经网络
- AdderNet是另一种替代乘法算法的技术，但允许大于1位的参数；用加法替换了所有卷积

$$Y(m,n,t) = -\sum_{i=0}^{d}\sum_{j=0}^{d}\sum_{k=0}^{c_{in}} |\mathbf{X}(m+i,n+j,k) - \mathbf{F}(i,j,k,t)|$$

- NAS可应用于BNN的构建

- 其他方法——主要减少参数的存储成本

  k-means：将权值或者激活值进行聚类

  HashNet：每个哈希组都被替换为一个浮点权值

  Huffman编码

  Hession方法：将聚类参数的平均hession加权量化误差最小化

  权值正则化（l2）可以通过惩罚大幅度的权值来略微提高量化网络的精度

  BN：有线性和非线性运算，解决内部协变量移位问题，也是一种典型的量化方法，非线性的运算可能会有数值不稳定性的问题（当使用线性表示时，低位表示可能会出现问题）；**改进**：$l_1 - norm$ BN（在正向和反向训练中只有线性运算）

- **Quantization-aware Training**：量化感知训练
  - 大多数量化方法使用全局（分层）量化来将全精度模型简化为一个简化的位模型，因此，可以导致不可忽略的精度损失。
  - 不可逆精度损失
  - BNN训练

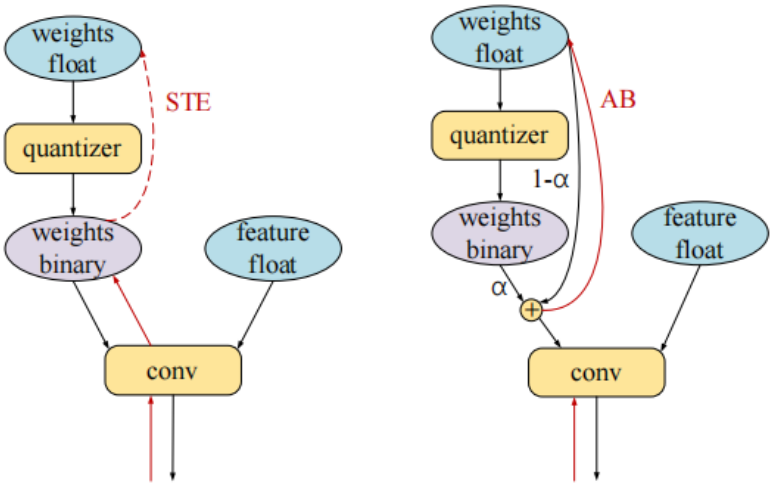
    对于具有二进值权值的二值化网络，由于通常的小导数，使用梯度体面的方法更新权值是不有效的。早期的量化网络是用一种名为**期望回归传播**（EBP）的贝叶斯推理来训练的
    - 训练：

1. EBP：分配有限的参数精度的权重和激活（例如，二值化的权重和激活）；EBP通过更新权值上的后验分布来推断具有量化权值的网络。通过微分反向传播的参数来更新后验分布。
2. **二元连接**采用了EBP的概率思想，但BC没有优化权值的后验分布，而是保留了浮点权值进行更新，然后将其量化为二进制值。
3. STE：

$$\text{Forward}: w_b = \text{sign}\left(w_r\right)$$

$$\text{Backward}: \frac{\partial c}{\partial w_r} = \frac{\partial c}{\partial w_b} \mathbf{1}_{|w_r| \leq 1}$$

4. Alpha混合（AB)可以作为STE的替代品：设置了一个超参数α；STE直接绕过量化器，而AB通过引入额外的系数剜来计算实值权值的梯度



5. DoReFa：

$$f_w^k = 2\,\text{quantize}_k\left(\text{limit}(w_r)\right) - 1$$

$$\text{where } \text{quantize}_k(w_r) = \frac{1}{2^k - 1}\,\text{round}\left(\left(2^k - 1\right) w_r\right), \quad (29)$$

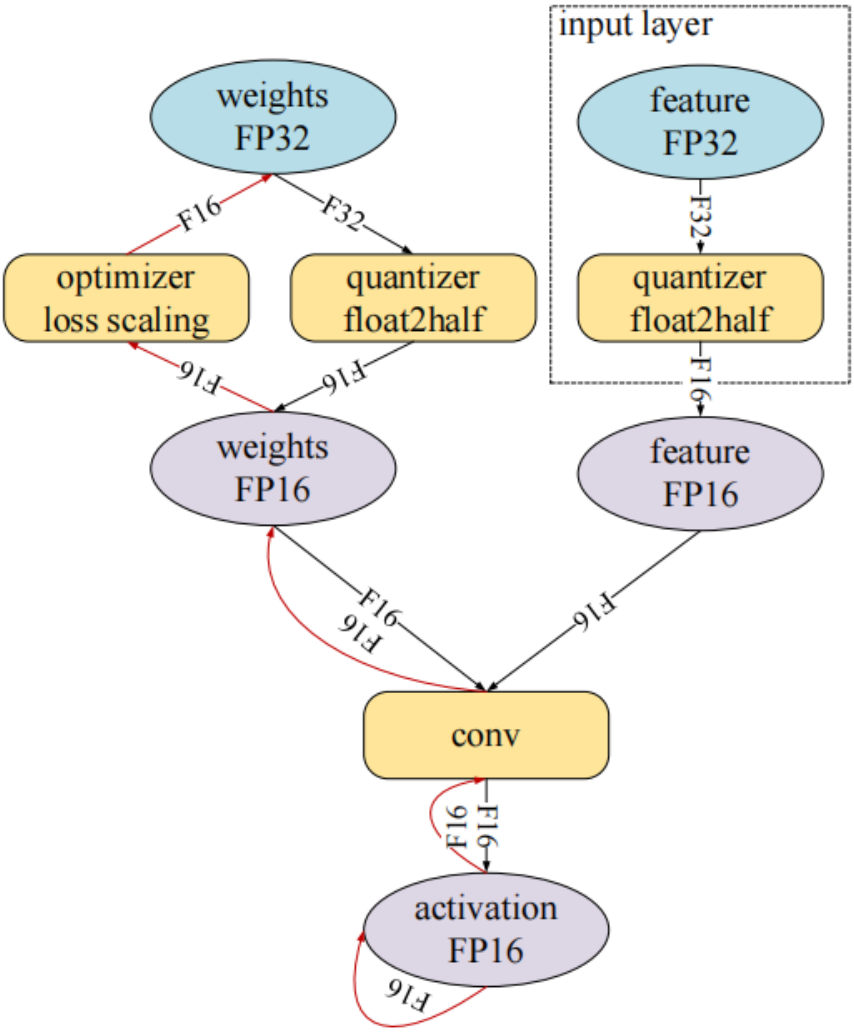
$$\text{and } \text{limit}(x) = \frac{\tanh(x)}{2\max(|\tanh(x)|)} + \frac{1}{2}$$

$$\tilde{f}_\gamma^k = 2\max_0(|dr|)\left[\text{quantize}_k\left(\frac{dr}{2\max_0(|dr|)} + \frac{1}{2}\right) - \frac{1}{2}\right]$$

解决梯度爆炸：将二值化对网络损失的影响表述为一个优化问题，由具有对角Hessian近似的近端牛顿算法解决，该算法直接使二值权值的最小化损失

- 低数值精度训练

  低数值精度的训练包括将低精度值进行正向和向后传播，同时保持全精度的累积结果；全精度权值保持梯度更新，而其他操作数使用半浮点。



总结

- 剪枝

  本文主要将剪枝技术分为动态剪枝和静态剪枝，其中动态剪枝是目前关注的焦点。

  剪枝可以通过多种方式执行：元素级修剪改进了权重压缩和存储；通道级和形状级剪枝能够被特定的硬件和计算库加速；过滤器级和层级能够显著减少计算复杂度；

  剪枝可以看作是NAS的一个子集，但搜索空间较小，尤其当剪枝后的权重可以不依赖于剪枝前的网络时。

  一些NAS技术也可以应用于剪枝方法，包括借用训练系数和强化学习搜索。

  如何进行有效的修剪：

  - 均匀的剪枝会引入精度损失，因此设置剪枝比按层变化会更好
  - 动态修剪可能导致更高的精度和保持更高的网络容量，首选动态剪枝
  - 从结构上修剪网络可能受益于成熟的库，特别是在高级修剪时
  - 再训练模型可能会比微调剪枝后的模型取得更好的结果
  - 基于幅度的剪枝比基于惩罚项的剪枝效果要好

- 量化

  本文主要介绍了二值化神经网络、低精度神经网络以及训练方法

  量化通常会因为信息丢失而导致精度损失，在紧凑的网络中表现尤其明显；

  大多数早期的低位量化方法只在小数据集上比较性能；非均匀分布数据可能会导致量化性能的进一步恶化，有时这可以通过微调的归一化或非线性量化来改善

8位量化作为精度和压缩之间的良好权衡，在实践中被广泛应用。它可以很容易地部署在当前的处理器和定制硬件上。

如何进行有效的量化：

- 使用不对称量化。它在量化范围内保持了灵活性，即使它有计算开销
- 量化权重而不是激活值，因为激活值对于数值精度更敏感
- 不量化偏差，偏差占内存空间少且能保留更多网络信息
- 按通道级进行量化
- 微调量化后的模型，能够减少精度损失
- 最初使用一个32位浮点模型进行训练。低比特量化模型很难从头开始训练，特别是在大规模数据集上的紧凑模型
- 量化的灵敏度排序：梯度、激活值、权重
- 训练量化模型时，进行梯度随机量化

未来工作

- 自动压缩

  自动量化是一种自动搜索量化编码来评估精度损失与压缩比的技术。

  自动修剪是一种自动搜索不同的修剪方法来评估稀疏度与准确性的技术。

  类似于超参数调优，可以在不进行人工干预情况下进行搜索

- 对其他类型的神经网络的压缩

  目前的压缩方法主要针对于CNN分类任务；未来的工作还应考虑其他类型的应用程序，如目标检测、语音识别、语言翻译

- 硬件适应

  硬件实现会限制剪枝算法的有效性。当在GPU [264]上使用im2col-gemm时，元素级剪枝只会略微减少计算或带宽，形状级修剪通常不能在专用的CNN加速器上实现。应考虑对硬件加速器进行压缩技术的硬件-软件协同设计，以实现最佳的系统效率。

- 全局方法

  目前的优化方法都是单层进行的优化，在优化过程中无法利用前后层的信息，最近，人们提出了考虑多层优化有效性的方法。讨论了剪枝结合张量分解，从而得到更好的整体压缩。

总结

修剪删除了对结果贡献有限的冗余计算。量化化通过降低数据类型的精度来减少计算。两者都可以单独使用或联合使用，以减少存储需求和加速推断。