

Pruning in Deep Compression

Jinze Chen

2019 年 1 月 19 日

1 Introduction

对网络进行剪枝主要有两个目的: 一是降低网络的内存与计算资源消耗以便于将其部署在资源有限设备上, 二是减轻过拟合现象, 因此剪枝可能提高某些网络精度。

对于后一种情况, 其整个流程类似于哺乳动物脑的发育过程, 在这过程中神经元的突触数量在儿童期间到达顶峰, 而后逐渐减少。然而本文并不对该内容进行阐述, 我们的最终目标是以更少的资源消耗尽可能地逼近目标网络。本文罗列了一些其他人在这方面做的努力, 将其集合起来, 并总结出网络剪枝必要的流程以及其注意事项。

2 Related work

在网络剪枝中最关键的问题是如何判别网络中不重要的成分并将其舍弃, 其中对计算资源以及网络精度的权衡也是很重要的一部分, 像是以权重为单元的剪枝通常精度较高, 但计算较慢, 相比之下更高级别的如层级别的剪枝则正好相反。还有一些通过改变原网络结构来降低存储量的做法, 具体实现将在3.4里描述。

总结起来越不重要的元素被舍弃时对网络的影响越小, 基于该原则韩松在 [1] 中提出了一种迭代剪枝的方法来压缩网络, 在这过程中绝对值小于一阈值的权重全被舍弃。实际应用时可以通过 l_1 与 l_2 正则化来更方便地实现。这种方法的主要缺陷是缺少通用性与灵活性。

为了克服上述缺点, 也有些将注意力转移到了“组”层面的稀疏性。Lebedev 与 Lempitsky 在他们的论文 [3] 中通过对损失函数的正则化来去除

一些权重组。与此类似, Wei Wen 在其论文 [11] 中提出了 Structured Sparsity Learning (SSL) 方法来正规化 filter, channel, filter shape 以及 depth structure。但是在该过程中原有网络结构已被损坏, 其结果是需要特殊的函数库来加速网络。

也有一些人以 filter 层面的剪枝为基础, 其研究关键点在与衡量神经元重要性, 这已经在 [2,5,7,9,12] 中被广泛研究了。最简单的方法是单纯根据其绝对值来衡量, 其中 Hao Li 在其论文 [5] 中以绝对和为衡量标准。还有的比较实用的方法是通过根据通过 ReLU 函数后的激活率来判断, 如 Hengyuan Hu 在 [2] 中的研究成果。这两种方法直接且简洁, 但它们并未将操作与结果直接联系起来。受此启发, Pavlo Molchanov 在其论文 [7] 中使用泰勒级数展开来分析移除 filter 对最终损失的影响。

3 Different methodologies

3.1 Neuron Pruning for Compressing Deep Networks using Maxout Architectures [8]

本论文提出了一种剪除整个神经元的方案, 在该操作后可以进一步地进行权重剪枝。

该方案主要是是哟你个了 maxout 单元以及其 model selection 能力来剪掉整个神经元, 从而压缩整个网络。该过程假设 maxout 单元中存在冗余, 具体流程如下图:

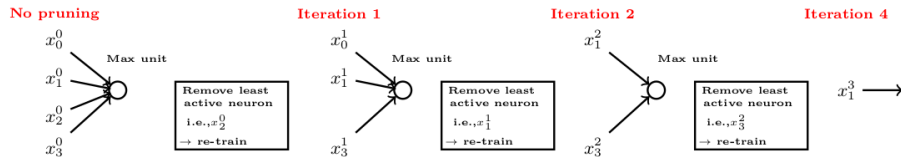


Fig. 2: Neuron pruning process for $k = 4$ inputs per maxout unit. x_a^b represents a neuron with ' a ' the neuron index and ' b ' the iteration.

首先训练一个包含 maxout 层的 CNN。该 maxout 层在 k 个相继的神经元间取最大值, 连接数得以减少 k 倍, 因此将该 maxout 层置于最多权重的曾会得到较好效果。再来, 通过记录某一神经元在 maxout 单元中被激活的次数, 激活次数较小的神经元被移除, 此时该神经元对整个网络的影响可以忽略。最后该 CNN 被重新训练, 训练完成后重复上述过程。

权重剪枝中绝对值较低的权重被忽略。该过程可以以其他准则替代。

仅进行神经元剪枝的情况下, LeNet-5 的网络体积最多降低了 74%, VGG16 降低了 61% 而网络表现不降低。

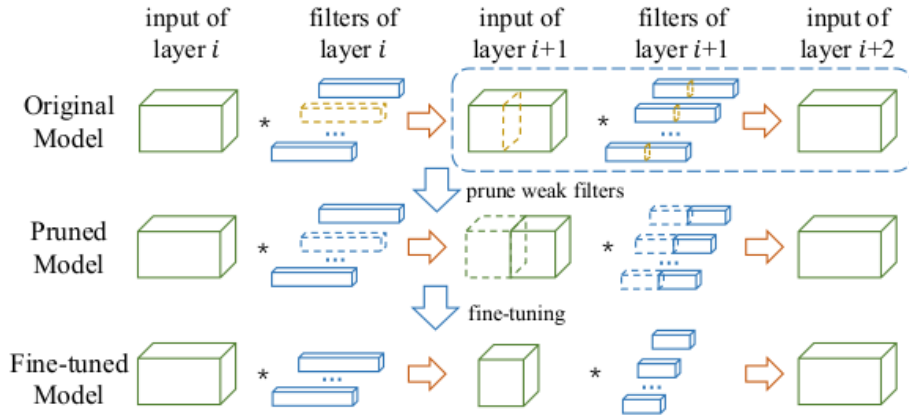
该论文主要的创新点是考虑 maxout 结构的冗余并对其处理。因为该过程可以与其他权重剪枝方法结合, 对于具有某些结构的网络可以考虑使用该方案。

3.2 ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression [6]

该论文提出了 filter 层面的网络剪枝方案, 被称为 ThiNet。使用这种方式可以在训练与推测阶段加速与压缩原网络。

本论文主要思路是将 filter 剪枝视为一优化问题, 即在 $i+1$ 层的 channel 中找到一子集来近似模拟 $i+1$ 层的输出。相对于考虑该舍弃哪些 filter, 作者考虑的是保留哪些重要 filter。

该方法对一已训练好的模型根据设定好的压缩率逐层进行进行剪枝, 整个过程示意如下:



1. 随机选出一样本集。
2. 使用贪婪算法来选择 channel, 以最小化

$$\arg \min_T \sum_{i=1}^M \left(\hat{y}_i - \sum_{j \in S} \hat{x}_{i,j} \right)^2 \quad s.t. \quad |S| = C \times r, S \in \{1, 2, \dots, C\}$$

3. 对各 channel 加权来最小化误差, 定义如下 $\hat{w} = \arg \min_w \sum_{i=1}^M (\hat{y}_i - w^T \hat{x}_i^*)^2$
4. 迭代至步骤一来对下一层剪枝。

作者使用这种方式在 VGG-16 上以 0.52% 的 top-5 准确度下降换来了 $3.31\times$ 的 FLOPs 下降以及 $16.63\times$ 的压缩率。

本文主要创新点是作者将剪枝问题视为一优化问题, 其中各 filter 的重要性以其到下一层的输出来衡量而不是本层。并且由于剪枝是在 filter 层面进行的, 计算复杂度与权重剪枝相比相对较低。

3.3 Sparsifying Neural Network Connections for Face Recognition [10]

该论文主要内容是针对人脸识别应用设计一个新的网络, 并在网络训练完成后对该网络进行剪枝操作来。对我们来说有用的只有其剪枝内容。

作者计算神经元间相关性, 并舍弃相关性较低神经元间的链接。对与全连接层等权重不共享的层, 给定一神经元 a_i 与其前一层中 K 个相连的神经元 $b_{i1}, b_{i2}, \dots, b_{iK}$, a_i 与 $b_{ik}, k = 1, 2, \dots, K$ 间的相关为:

$$r_{ik} = \frac{E[a_i - \mu_{a_i}][b_{ik} - \mu_{b_{ik}}]}{\sigma_{a_i} \sigma_{b_{ik}}}$$

其中 $\mu_{a_i}, \mu_{b_{ik}}, \sigma_{a_i}, \sigma_{b_{ik}}$ 分别代表 a_i 与 b_{ik} 的均值与标准差, 其值在另一训练集中获得。对于正和负的相关值, 作者使用了不同的方式处理。对于卷积层作者使用另一公式来计算, 在这里我们不展现。

本论文主要的创新点是根据层间关系权衡重要性。除此之外, 作者也尝试了从头开始训练一稀疏网络, 结果在人脸识别上效果并不好。尽管我们的关注点是对已知网络进行剪枝, 该结论仍具有很强的指导意义。

3.4 DeepRebirth: Accelerating Deep Neural Network Execution on Mobile Devices [4]

本论文找到了一种提高模型预测速度与减小模型体积的方法, 被称为 DeepRebirth。严格来说这一方法不应被称为剪枝, 但是它对于在移动端的部署是十分有用的, 并且作者在实际的手机上测试了这一结果, 因此我们把这种方法列了出来。

首先作者发现非张量层在运行中消耗很多时间, 如下图:

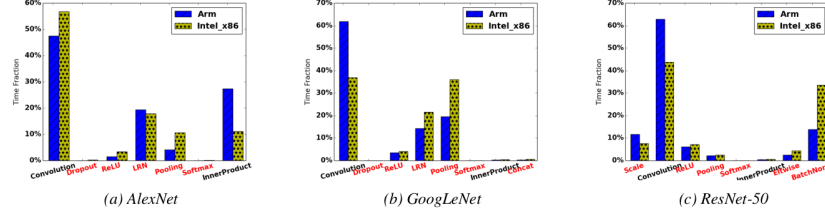


Figure 2: Time Decomposition for each layer. Non-tensor layers (e.g., dropout, ReLU, LRN, softmax, pooling, etc) shown in red color while tensor layers (e.g., convolution, inner-product) shown in black color.

因此作者想要通过将非张量层与张量层合并为一层来提高运行效率。这篇文章提出了两种合并方法, 分别为 **Streamline Slimming** 和 **Branch Slimming**。

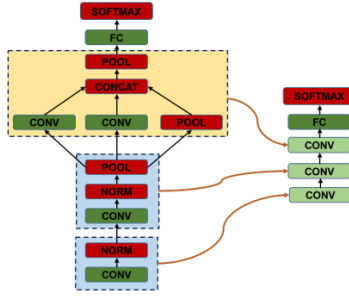


Figure 1: An illustration of proposed DeepRebirth model acceleration pipeline. DeepRebirth optimizes a trained deep learning model (left) to an accelerated "slim" model (right). Such optimization is achieved with two operations: Streamline Slimming which absorbs non-tensor layers (i.e., pooling and normalization) to their bottom convolutional layer (in light blue background) and Branch Slimming which absorbs non-tensor branches and convolutional branches with small convolution filters (e.g., 1x1) to a convolutional branch with large convolution filters (e.g., 5x5) (in light yellow background). We name new generated layers as slim layers.

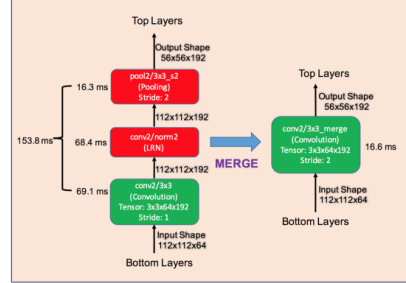


Figure 3: Streamline Slimming: The GoogLeNet example and the running time is measured using bvlc_googlenet model in Caffe on a Samsung Galaxy S5. Left panel: convolution (in green), LRN (in red), pooling (in red). Right Panel: single convolution layer. The three layers in the left panel are merged and regenerated as a convolution layer (i.e., slim layer) in the right panel.

权重与偏移经过重训练来获得与原网络相似的效果, 如下式:

$$(\tilde{W}^*, \tilde{B}^*) = \arg \min_{W, B} \sum_i ||Y_{CNN}^i - \tilde{f}(W, B; X^i)||_F^2$$

where W and B represent weight and bias matrix respectively. 其中 W 和 B 分别代表权重与偏移矩阵。

本文主要的创新点是分析得到了非张量层的运行时间很长并通过合并特定层解决了该问题, 然而该方式完全为工程方向, 并不具有太多理论指导意义。好处是这一做法可以用于任何训练后的网络部署中。

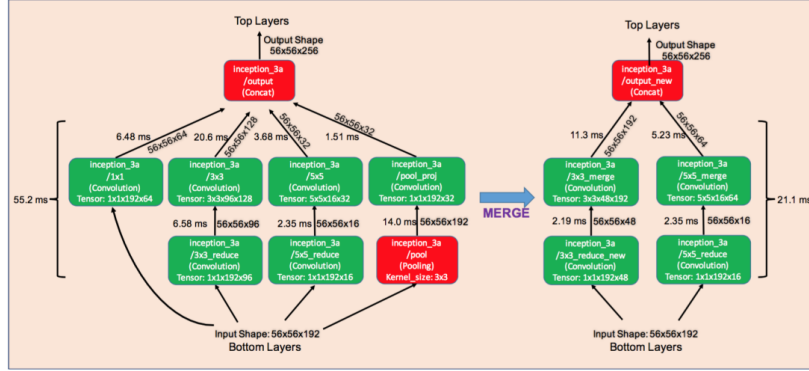


Figure 4: Branch Slimming: The GoogLeNet example and the running time is measured using `bvlc_googlenet` model in Caffe on a Samsung Galaxy S5. Left panel: four branches in parallel, convolution layer, convolution + convolution, convolution + convolution, convolution + pooling. Right panel: two branches in parallel, convolution + convolution, convolution + convolution. Two branches are reduced.

4 Summary

这些论文都提出了一些加速或压缩神经网络的方案。大体上来说，我们可以得出对一已知网络剪枝流程如下：

1. 获取一训练好的神经网络。
2. 根据其组成元素重要性进行剪枝，其中通过层间重要性来判断通常能获得更好的效果。
3. 对该网络进行重训练来最小化剪枝误差，然后回到步骤 2。

*4 重构一网络来降低模型延迟。

为了进行步骤 4，我们需要在其他一些方面入手，或者我们只关注模型速度或模型大小。无论如何，上述过程应该就是网络剪枝的大体流程，而在整个过程中我们还可以采用如量化与熵编码等错吃来进一步压缩模型。

参考文献

- [1] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.
- [2] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016.

- [3] Vadim Lebedev and Victor S. Lempitsky. Fast convnets using group-wise brain damage. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2554–2564, 2016.
- [4] Dawei Li, Xiaolong Wang, and Deguang Kong. Deeprebirth: Accelerating deep neural network execution on mobile devices. In *AAAI*, 2018.
- [5] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.
- [6] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5068–5076, 2017.
- [7] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, abs/1611.06440, 2016.
- [8] Fernando Moya Rueda, Rene Grzeszick, and Gernot A. Fink. Neuron pruning for compressing deep networks using maxout architectures. In *GCPR*, 2017.
- [9] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.
- [10] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Sparsifying neural network connections for face recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4856–4864, 2016.
- [11] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016.

- [12] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016.